Making additions to the hyphenation dictionary

Here are the instructions for making additions to the hyphenation dictionary. 1

Make an additions file 1a

Use the Create Command in the HYPHEN subsystem to process your documents to extract words that are not in the current hyphenation dictionary. Use the Insert command in the HYPHEN subsystem to hyphenate the words in your additions file and the Modify command in the HYPHEN subsystem to correct the hyphenation. For details, use the Help command with 'hypen' as the phrase to look up. 1a1

Make a new hyphenation dictionary 1b

Use the Add command in the HYPHEN subsystem to create a new version of the dictionary file. This command requires a dictionary file, which is a '.txt' file, and an additions file, which is an '.his' file. 1b1

For the dictionary file use 1b1a

(xporgen, hyphen-dictionary.txt,) 1b1a1

For the additions file use your additions file. 1b1b

The Add command will create a new version of the dictionary file (xporgen, hyphen-dictionary.txt,). 1b2

Update the Output Processor hyphenation SAV file 1c

Use the Runfile command from the EXEC to update the Output Processor hyphenation SAV file. The name of the runfile that does this update is <xporgen>hycreate.txt. This runfile reads the new version of <xporgen>hyphen-dictionary.txt and creates a new version of <netsys>hyphlook.sav, which is the SAV file used by the Output Processsor for hyphenation. 1c1

must have write privlages

@runfil.sav;1

runf(esc) (not run esc)

Konitar, Kendler

```
(J70319) 12-May-78 16:25;;;; .HJOURNAL="RICH 12-May-78 16:25  70319";
Title: .H1="INFORMATION.NLS 10.0 Information File, Release 2.0";
Author(s): Richard W. Zellich/RICH; Distribution: /DRXAL-HD( [ ACTION ]
) ; Sub-Collections:  NIC DRXAL-HD; Clerk: RICH;  .IGD=0; .SNF=HJRM;
.RM=HJRM-7;  .PN=-1; .YBS=1; .PES; Origin: < ELITE, INFORMATION.NLS;11,
>, 12-May-78 07:45 RICH ;;;;####;


   The following tools are available for your use:

      ELITE tools:
         Calendar, PM-regulations

      Utility tools:
         Base, Calculator, Format, <>Ftp, Graphics, Message, Programs,
         Publish, Sendmail, <>Spell, Useroptions

      "NAMED" tools [Goto (tool) Named...]:
         Any other tool known to you (if a NAMED tool is not in either
         your directory or the NETSYS directory, you must type:
         directory-name,tool-name)

      EXEC:
         "Goto (tool) EXec" will take you to the executive operating
         system (TENEX or TOPS-20) in a lower fork.
   NEWS
      No news - "Show <>Old-news" will display all past "news" items.
   OLD-NEWS
      No Old News - The system is in it's initial configuration.
```

< CJOURNAL, 70388.NLS.1, >, 16-May-78 19:29 XXX ;;;; .HJOURNAL="ROM 16-May-78 16:54 70388"; Title: .H1="Structuring the CML of universal commands"; Author(s): Raphael Rom/ROM; Distribution: /ARC-DEV([ACTION]) ANDY([ACTION]) ; Sub-Collections: NIC ARC-DEV; Clerk: ROM; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;

The current structure of the CML code for the universal commands is structured in an "all or nothing" way i.e., you cannot include portions ot a command syntax. In some cases it may be necessary to use only portions of the syntax and thus, with good structuring, INCLUDEs can (and should) be used.
I suggest to structure the commands such that it consists of a branch the head of which is the command statement with the first command word and undernearth it the actual rule plus all the other rules that are necessary.
Here is an example of the jump command:

```
   jump COMMAND = "JUMP"
      <"to"> jumprule;

      jumprule =
         ( IF DISPLAY
            ( NULL
               ent _ #"STATEMENT"
               dest _ DSEL( #"LOCATION" )
               vs _ viewspecs()
               CONFIRM
               xjump( ent, dest, vs, WINDOW )
            / "ADDRESS" <"relative to">
               ent _ #"STATEMENT"
               dest _ DSEL( #"STATEMENT" )
               <"modified by address">
               addr _ LSEL( #"ADDRESS" )
               vs _ viewspecs()
               CONFIRM
               xjumpaddr( #"STATEMENT", dest, addr, vs, WINDOW )
            )
         / IF NOT DISPLAY
            "ADDRESS"
               ent _ #"STATEMENT"
               dest _ DSEL( #"CHARACTER" )
               vs _ NULL
               CONFIRM
               xjump(ent, dest, vs, WINDOW)
         / jmpcoms
            CONFIRM
            xjump( ent, dest, vs, WINDOW )
         / jmpret
         / "CONTENT"
            ( "FIRST"
               ent _ #"FIRSTCONTENT"
            / "NEXT"
               ent _ #"NEXTCONTENT"
            )
               %display previous content%
               ( IF lastcontent _ xjmpcnt(#"CONTENT") SHOW ( lastcontent
      ) RPT dest _ FALSE %for repeat%
         / dest _ LSEL(#"TEXT"))
      vs _ viewspecs()
   CONFIRM
   xjump( ent, dest, vs, WINDOW )
/ "WORD"
   ( "FIRST"
      ent _ #"FIRSTWORD"
   / "NEXT"
      ent _ #"NEXTWORD"
   )
      %display previous content%
      ( IF lastcontent _ xjmpcnt(#"WORD") SHOW ( lastcontent )
      RPT dest _ FALSE %for repeat%
         / dest _ LSEL(#"WORD"))
      vs _ viewspecs()
   CONFIRM
   xjump( ent, dest, vs, WINDOW )
);

jmpcoms =
   ent _
      ( "ITEM" ent _ #"STATEMENT"
      / "SUCCESSOR"
      / "PREDECESSOR"
      / "UP"
      / "DOWN"
      / "HEAD"
      / "TAIL"
      / "END" <"of Branch">
      / "BACK"
      / "ORIGIN"
      / "NEXT"!L2!
      ) dest _ DSEL(#"CHARACTER")
      vs _ viewspecs()
   /
      ( ent _ "LINK"
         dest _ LSEL(#"LINK") vs _ NULL
      / ent _ "NAME"
         ( "ANY"
            ent _ #"NAME"
         / "FIRST"
            ent _ #"FIRSTNAME"
         / "NEXT"
            ent _ #"NEXTNAME"
         / "EXTERNAL"
            ent _ #"EXTNAME"
         )
            dest _ LSEL(#"NAME") vs _ viewspecs()
      );
jmpret =
   ent _ "RETURN"
      CONFIRM
      dest _ xgetjumpring( ent, WINDOW )  %get statements%
      pfjmpi(FALSE)  %initialize index for statement array%
      jmpring
```

```
            / ent _ "FILE"
               ( "NAMED"
                  ent _ #"FILENAMED"
                  dest _ LSEL(#"OLDFILENAME")
                  vs _ viewspecs()
                  CONFIRM
                  xjump( ent, dest, vs, WINDOW )
               / "RETURN"
                  ent _ #"FILERETURN"
                  CONFIRM
                  dest _ xgetjumpring( ent, WINDOW )  %get file names%
                  pfjmpi(FALSE)  %initialize index for file name array%
                  jmpring

               );
               jmpring =  %display strings from array until user says
               OK%
                  SHOW (#"
                  ")  %go to new line%
                  SHOW (pfjmps(dest))
                  param _ ANSWER
                  (IF param xjumpreturn(ent, pfjmpi(TRUE), WINDOW)
                  / IF NOT param jmpring);
```

One example where it may be useful is when a subsystem needs to take some extra precautions when a user performs a jump command. This can be programmed in the following way:

```
   specialjump COMMAND = "JUMP" <"to">
      specialstuff
      INCLUDE jumprule
      morespecialstuff;
```

< CJOURNAL, 70475.NLS.1, >, 26-May-78 18:31 XXX ;;;; .HJOURNAL="RLL 26-May-78 16:34 70475"; Title: .H1="List of Architects and quasi-architects with the Client Coordinator"; Author(s): Robert N. Lieberman/RLL; Distribution: /ARC([INFO-ONLY]) ; Sub-Collections: NIC ARC; Clerk: RLL; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;

.PEL; .PN=PN-1; .GCR;Hope there are no surprises here.

Organization :	Architect :	Ident :	CC :
AFDSDC	Carpenter		
AFSC			DLS
ALMSA	Zellich	RICH	PKA
ARPA	McLindon		RH
AVRADCOM	Ball		PKA
BRL	Taylor	SMT	DLS
DARCOM-DMIS	McCarty		DLS
DMA			DVN
ELITE	Zellich		PKA
LSSA	McCarty		JMB
NAVLEX	Schill	JOHN	RH
NSA	Barney		RH
NIC	Feinler		
NSF	Custer	PAUL	RLL
NSRDC	Avrunin	ILA	PKA
PTFD	McCarty		JMB
RADC	Kennedy		JMB
SRI			
TECOM			JMB
TRW	Peterson		DVN
TYM			BJS
YUMA			JMB
WEYER	Roberts		EKM

< CJOURNAL, 70581.NLS.1, >, 6-Jun-78 13:01 XXX ;;;; .HJOURNAL="DLS 6-Jun-78 07:33 70581"; Title: .H1="Attaching Machines to the ARPANET"; Author(s): Duane L. Stone/DLS; Distribution: /JAKE([ACTION]) ARC([INFO-ONLY]) ; Sub-Collections: ARC; AccessList: JAKE ARC DLS; Clerk: DLS; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;

.PEL; .PN=PN-1; .GCR;Jake, since you have recently returned from the sponsor's group meeting, thought maybe you could add facts to the rumors.

It looks like RADC will sponsor the KI's connection to the ARPANET under the PLP. I believe we have plans to attach other machines in the future, either at TYM or at customer's sites. This may not be possible, however. Apparently Honeywell has very quietly said that they will no longer support (after July) the minis that are the backbone of the ARPANET (H-516/316). Some of the host interface equipment that is considered a part of the IMP/TIP is made by Honeywell. DCA has asked that all requests for node upgrades be sent to them before 16 June 78. If parts of host interfaces are no longer made, then our plans to connect to the ARPANET in the future may go down the tubes, regardless of our ability to find sponsors. It's hard for me to believe that DCA has not made contingency plans, perhaps with BBN, to supply host interface equipment in the future. There must be many sites that are planning on attaching more machines. DCA can't expect everyone to upgrade to a PLURIBUS, can they?

Jake can you shed any light on this? I talked briefly with Tom Lawrence after the sponsor's group meeting, but he couldn't add much. If this is really the case, why hasn't DCA sent out numerous warning messages?...at least I haven't run across any.

I'm also curious about the status of software in TENEX and TOPS20 machines for the extended header/leader. Is ARPA or DCA or anyone sponsoring BBN to do this. Will DEC do it on their own for TOPS20? when?

I'm also curious about the "legal" implications of offering a commercial service over the ARPANET. DoD clients' use of AUGMENT may be experimental, but the system is billed as a commercial product. Can we reasonably expect to mix DoD and commercial clients on the same machine as SRI did in the past? Is there precedent for this? Are there other sites (BBN or SRI?) that deliver service to DoD and commercial clients on the same machines?

I also wonder to myself occasionally, who owns the "TYM-IMP"? Isn't it ARPA? What if they decided they needed the IMP elsewhere? Chances are slim they would risk ticking off the Army or AF by pulling it out, but stranger things have been known to happen.

< CJOURNAL, 70864.NLS.1, >, 1-Jul-78 16:30 XXX ;;;; .HJOURNAL="DLS 30-Jun-78 12:07 70864"; Title: .H1="Some Pricing Examples for AUGMENT"; Author(s): Duane L. Stone/DLS; Distribution: /RC3([INFO-ONLY]) EJK([INFO-ONLY]) JLM([INFO-ONLY]) ; Sub-Collections: ; Clerk: DLS; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES;

.PEL; .PN=PN-1; .GCR;Re: (70863,)

Re: the new price schedule for AUGMENT services (70863,) in FY-79...AUGMENT = Workshop Utility Services in this context. It may be a little confusing, as it was to me when I first saw it. The translation between the old Computer Resource Units (CRU) and the new Service Units (SU) is not immediately obvious. Let me try with a couple of examples to clear up any confusion that might remain.

In FY-78 RADC purchased 14 CRUs. This is the equivalent of 14 / 3 (3 CRUs per display) or 4.67 people working full time at display terminals for 12 months.

To translate this into SUs; it is the equivalent of 4.67 X 4 Login Units (LU) X 2 (resource level) X 12 months, or 448 SUs. Looking in the table (70863,11:g), the FY-79 price for that level of SUs is $260; giving a total of 448 X $260 = $116480...or about 58% of what you paid in FY-78 ($200,000).

Looking at it another way, if you have $200,000 to spend in 79, this is 800 SUs at $250 each. It takes 96 SUs to support a display working 8 hours a day for 12 months, hence $200,000 will allow 8.3 users to be logged on full time, or about 8.33/4.66 = 1.78 times the service for the same amount of coins.

As you can see from the table (70863,11:g), if one contracts for a more SUs (either at a higher level or for a longer time) the price drops substantially. For example, if AFSC adds their money to RADC's, the result is that both parties could get an additional price break. You will also note that training has been completely unbundled, so that you buy only what you need.

If none of this makes sence, I'd be glad to stop by and work through some examples with you.

Stoney

The whole point of using a CRT with DNLS is to USE THE MOUSE. If you want to use an existing CRT without a mouse, then the thing to do is to use TNLS.

TNLS is pretty damn good (except for actions like Transpose, where you REALLY need to point to things) and I find it far superior to ANY sequential editor I have ever been exosed to.

Essentially, a Line-Processor IS CRTSTY code, only moved out to a cheap microprocessor to save the user the expense of buying a bigger slice on the more-expensive PDP-10 to support the extra overhead. Also, this means the guy using DNLS pays for the overhead (he bought the Line-Processor) instead of every user on the host paying a share of the CRTSTY overhead.

I'd rather use non-DNLS CRT'S with TNLS than play with a hack like "mouseless DNLS", with it's super-clumsy cursor control (except in EXTREME cases), and would certainly consider buying a Line-Processor custom-programed for whatever terminal I already had, before I'd go off and abandon NLS in favor of any kind of sequential editing system.

```
< ZELLICH, A1-CHARTS.NLS;2, >, 13-Apr-80 17:20 RICH ;;;;   Title:
Author(s): Richard W. Zellich/RICH; Distribution: /BILL( [ INFO-ONLY ] )
JPM2( [ INFO-ONLY ] ) GFR( [ INFO-ONLY ] ) PL2( [ INFO-ONLY ] ) DAB2( [
INFO-ONLY ] ) IMP( [ INFO-ONLY ] ) LJC( [ INFO-ONLY ] ) ;
Sub-Collections:  NIC; Clerk: RICH;          Origin: < ZELLICH,
A1-CHARTS.NLS;13, >, 6-Nov-79 12:26 RICH ;;;;####;
```

Tymshare A1 Briefing Charts from HQ DARCOM Briefing

Suggested viewspecs for printing this document are <:wynCF>. 1

The following represents the charts (viewgraphs) Tymshare used to brief HQ DARCOM personnel in mid-September. Pages/charts 2, 4, and 5 were line drawings and could not be reproduced here. Comments enclosed in brackets ([ ]) are annotations made on the Xeroxed charts by one of the attendees (Bill Walsh?) from information presented by Tymshare. 2

PAGE 1 3

THE AUGMENT ENGINE 3a

Why build another computer? 3b

What is an A1? 3c

How much does an A1 cost? 3d

How much is the software? 3e

What are the benefits to DARCOM? 3f

What configuration is proposed for DARCOM's prototype? 3g

What is the A1's cost-effectiveness to DARCOM? 3h

PAGE 2 4

Figure 1: A1 System Module

This chart is a system schematic showing the hardware configuration, with bus interfaces, control and data paths, etc. It is a line drawing and cannot be reproduced here without using a graphics workstation. 4a

PAGE 3 5

NETWORK CAPABILITIES 5a

*LSI-11 based network processors (11/2: 1500 cps thruput, 11/23: 3000 cps thruput) 5b

*Economical packet network for specialized purposes (local nets, concentrators, etc.) 5c

*Gateways to ARPANET and TYMNET 5d

*X.25 Interface [currently operational] 5e

*Direct memory interface to a1 for efficiency 5f

*Remote printer support 5g

PAGE 4 6

This chart is an overall system schematic, showing the network connections, separate FE and BE/FS (FileSystem) computers, tapes, disks, etc. It is a line drawing and cannot be reproduced without using a graphics workstation. 6a

PAGE 5 7

This chart is a larger overall system schematic, depicting an AUGMENT environment very similar to the DARCOM Distributed Processing Architecture plan: Communications Processors connected to network(s) and user terminals, FE computers, BE computers, and FS (FileSystem) computers cross-connected to all file-system disks. It shows swapping disks on the BE computers but not on the FE's or CP's. It is a line drawing and cannot be reproduced here without using a graphics workstation. 7a

PAGE 6 8

A1 HARDWARE PRICES 8a

ITEM	PRICE	MONTHLY MAINTENANCE	8b
A1 Processor + 512K Words Memory	$80,000	$600	8c
160 MB Disk	9,100	75	8d
75 IPS Tape	5,290	50	8e
600 LPM Printer	11,245	150	8f
ARPANET Connection	14,300	100	8g
Add On Memory (256K)	18,750	175	8h
Communication Node (11/23)	6,500	55	
[either/or]			
Communication Node (11/2)	5,200	45	8i

Terminal Line Units	3.250	30	8j

Installation charges 3% of purchase price with a minimum of $500 for any single installation trip. 8k

PAGE 7 9

A1 SOFTWARE PRICES 9a

ITEM	PRICE	APPLIES	9b
TENEX	$ 20,000	Once per cluster [perpetual license]	9c
AUGMENT for singlet	120,000	Once per cluster	9d
Cluster support	100,000	For each additional processor	9e
UIS System only	80,000	Each processor [Reach-Thru only]	9f

Maintenance for purchased software is 10% per year. (First Year included in purchase.) 9g

Software licensed for specific cluster system. 9h

AUGMENT contains following software: 9i

BASE	- Basic word processing and core functions	9j
PROGRAMS	- Accessing additional subsystems	9k
OUTPUT PROCESSOR	- Formatting documents for terminal and printer	9l
MAIL	- New Electronic Mail Handler plus Journal	9m
PLUS	- Most existing subsystems currently depended upon by users (e.g. Xtable, Spell, Hyphen...)	9n

NEW MAJOR SUBSYSTEMS TO BE UNBUNDLED. 9o

PAGE 8 10

LEASE TERMS FOR AUGMENT 10a

MONTHLY LEASE RATES FOR AUGMENT HARDWARE: 10b

1YR	2YR	3YR	4YR	5YR	10c
.037	.034	.031	.028	.028	10d

Lease rate does not include hardware maintenance. 10e

MONTHLY LEASE RATES FOR AUGMENT SOFTWARE: 10f

.025 regardless of term. 10f1

Lease rate does not include software maintenance. 10g

PAGE 9 11

TERMINALS 11a

* AUGMENT 1200 11b

  * Jan 1, 1980 Price Reduction 11c

Purchase:	4500 => 3285		11d
1 yr. lease:	225/mo => 172/mo		11e
5 yr. lease:	- 126/mo		11f

* New AUGMENT Terminal Being Planned 11g

  - New Styling 11h

  - Greater expandability of firmware 11i

  - Target price under $3,000 11j

* Updated line-processor for interfacing to existing terminals 11k

PAGE 10 12

PROVEN TECHNOLOGY 12a

Hardware 12b

Bus oriented system 12c

Micro-Code technology 12d

16K MOS RAM chips - ECC 12e

P A G E 11 13

PROVEN TECHNOLOGY 13a

Software 13b

AUGMENT 13b1

over 15 years R&D 13b1a

over 5 years delivered service 13b1b

over 200 users 13b1c

TENEX 13b2

in widespread use on DEC 10 & 20 machines since 1971 13b2a

UNIX - not supported 13b3

P A G E 12 14

CAPITALIZE ON EXISTING INVESTMENTS 14a

Existing Applications - ELITE 14b

ELITE - Umbrella Subsystem 14b1

REGS - AUGMENT Subsystem 14b2

MILESTONES - AUGMENT Subsystem 14b3

CALENDAR - AUGMENT Subsystem 14b4

SUSPENSE - AUGMENT Subsystem 14b5

SCHEDULE - Reach-Through Tool 14b6

CPR - Reach-Through Tool 14b7

PROPERTY - AUGMENT Subsystem 14b8

PAGE 13 15

CAPITALIZE ON EXISTING INVESTMENTS 15a

Users Already Trained 15b

Over 6 PM offices currently using ELITE 15b1

2 or more DMIS shops already using ELITE 15b2

Other PM offices and DMIS shops want ELITE 15b3

In-House adhoc applications growing - EEO 15b4

PAGE 14 16

CAPITALIZE ON EXISTING INVESTMENTS 16a

* ALMSA Design, Development & Training staffs already trained 16b

* ARC Support Available 16c

PAGE 15 17

SAMPLE PROTOTYPE CONFIGURATIONS 17a

Singlet level - 15 simultaneous users (16 ports) 17b

ITEM	PRICE	ANNUAL MAINTENANCE	17c
1 A1	$ 80,000	$ 7,200	17d
2 Disks	18,200	1,800	17e
1 Tape	5,290	600	17f
1 Printer	11,245	1,800	17g
1 Communication Node	6,500	660	17h

1 TLU	3,250	360	
	$124,485	$12,420	17i

PAGE 16 18

SAMPLE PROTOTYPE CONFIGURATIONS 18a

Doublet Level = 40 Simultaneous Users (64 Additional Ports) 18b

ADD: 18c

ITEM	PRICE	ANNUAL MAINTENANCE	18d
1 A1	$ 80,000	$ 7,200	18e
2 Disks	19,200	1,800	18f
2 Comm Nodes	10,400	1,080	18g
4 TLU	13,000	1,440	
SUBTOTAL:	$122,600	$11,520	18h
TOTAL:	$247,085	$23,940	18i

PAGE 17 19

DARCOM LEASE PRICE FOR PROTOTPE 19a

Purchase price for prototype	$247,085	19b
Add ARPANET connection	14,300	
TOTAL HARDWARE PRICE	$261,385	19c
Monthly (1 yr) lease rate = 261,385 X .037	$ 9,671	19d
Monthly lease on software = 240,000 X .025	6,000	
	$ 15,671	19e
Monthly Hardware maintenance	2,095	19f

Monthly Software maintenance	2,000	
TOTAL MONTHLY PRICE	$ 19,776	19g
Annual price for software & hardware	$ 188,052	19h
Annual price for hardware maintenance	25,140	19i
Annual price for software maintenance	24,000	
TOTAL ANNUAL PRICE	$ 237,192	19j

PAGE 18 20

COST MODEL ASSUMPTIONS 20a

DOUBLET SYSTEM 20b

* 40 Simultaneous (peak) active users 20c
* 30 Simultaneous (average) active users 20d
* Amortize equipment over 5 years 20e
* Typical user active for 2 hours/day 20f
* 120 users (directories) supportable 20g

PAGE 19 21

A1 CLUSTER 21a

COST ANALYSIS 21b

ITEM	PURCHASE	5 YR LEASE	1 YR LEASE	21c
Annual Hardware Cost	52K	78K	116K	21d
Annual Software Cost	48K	72K	72K	
SUBTOTAL	100K	140K	188K	21e
Hardware Maintenance	24K	24K	24K	21f

Software Maintenance	24K*	24K	24K	
	----	----	----	
	148K	188K	236K	21g
Users Supported	120	120	120	21h
Cost/User/Year	$1,233	$1,566	$1,966	21i
Terminal Cost [Amortize over 5 years]	765	765	765	
	-------	------	-------	
Total Cost/User/Year	$1,998	$2,331	$2,731	21j

*No charge for first year 21k

Factors not included:

Operations - primarily simple tape archiving/retrieval
Facilities - Normal Office environment
Supplies
Training
Applications analysis and development
Network Communications 21l

```
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000111111111122222222223333333333444444444455555555556666666666777777777
23456789012345678901234567890123456789 0123456789012345678901234567890123456789
*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,
```

73447

```
RRRRRRRR        000000       MM       MM
RRRRRRRR        000000       MM       MM
RR      RR    00      00     MMMM   MMMM
RR      RR    00      00     MMMM   MMMM
RR      RR    00      00     MM  MM  MM
RR      RR    00      00     MM  MM  MM
RRRRRRRR      00      00     MM       MM
RRRRRRRR      00      00     MM       MM
RR  RR        00      00     MM       MM
RR  RR        00      00     MM       MM
RR    RR      00      00     MM       MM
RR    RR      00      00     MM       MM
RR      RR      000000       MM       MM
RR      RR      000000       MM       MM
```

```
*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,


*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,


*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,


*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,


*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,


*START* User ROM  Job  FTPSER Seq. 2366 Date  9-Nov-79 14:54:33 Monitor SRI-KL,
```

< MJOURNAL, 73447.NLS;1, >, 8-Nov-79 16:31 XXX ;;;; .HJOURNAL="JDH 8-Nov-79 14:35 73447"; Title: .H1="Line Protocol for the Augment 1200 Terminal"; Author(s): J. David Hopper/JDH; Distribution: /NPG([INFO-ONLY]) ARC([INFO-ONLY]) KWAC([INFO-ONLY]) ; Sub-Collections: NIC NPG ARC KWAC; Clerk: JDH; .IGD=0; .SNF=HJRM; .RM=HJRM-7; .PN=-1; .YBS=1; .PES; Origin: < HOPPER, A12PROTOCOL.NLS;15, >, 5-Nov-79 18:11 JDH ;;;;####;

.PEL; .PN=PN-1; .GCP;Technical document suitable as a reference for someone wishing to use the features of the Augment 1200 with applications programs other than Augment, or alternatively for implementing the protocol on another terminal so it may be used with Augment.

(protocol) Line Protocol for the Augment 1200 Terminal

Introduction

This document is a detailed description of the Augment 1200 line protocol. It is intended to serve as a guide to anyone wishing to implement the protocol on another terminal, as well as a piece of documentation for the terminal.

The scope of this document basicly covers terminal firmware supplied in EPROM and labeled "LPN4", although there are some references to special-purpose, down-line loaded terminal programs in common use.

This protocol is based on the Line Processor protocol described in (70398,). There are several additions, and some deletions. There are minor changes in the functioning of the terminal as compared to a Line Processor and terminal combination.

Additions include scroll window capability (to scroll an arbitrary rectangle up or down some number of lines with a single command), an improved printer driver protocol, and a down-line-load protocol for alternative terminal programs.

References to "positioning" and "tracking" in the protocol descriptions have been rewritten to reflect the separation of the mouse tracking function (bug) from the character writing position. It is no longer appropriate to refer to a "cursor" as used in conventional alpha/numeric displays.

The Augment 1200 terminal includes the capabilities of both types of Line Processors--it has both graphics and serial printer drivers. As with it's predecessor, the graphics protocol drives either a Tektronix 4012 or 4014 storage tube display.

Conventions

Conventions for this document

In this document, octal numbers are followed by "B".

"Unescorted" means that characters are sent as is without wrapping them in a protocol sequence.

Protocol Parameters

Most descriptive identifiers used in the protocol refer to variables which typically have a range [0,n] where n is usually < 93, but may be larger. Such a variable, "VAL" for instance, will generally occur as "VAL′" (VAL prime) in the protocol. This signifies that it is either sent as a single character (VAL+40B) if VAL is less than 93, or as a three character sequence

COORDESCAPE, VAL1, VAL2

where

COORDESCAPE=36B

VAL1 = (VAL high order six bits) + 40B

VAL2 = (VAL low order six bits) + 40B

(the term "COORDESCAPE" is used since this form is usually only nescessary in sending coordinates)

The "prime" (′) is used as a reminder that 40B has been added, or that the three character format might be used.

Coordinates

Terminal Screen Coordinates

Coordinates designate character positions. For example (1,1) is the second character on the second line up from the bottom.

The origin is at the lower left corner of the screen.

As a component of the protocol, the term "coordinates" denotes the string of characters which define a screen position. They are usually written "X′, Y′" with X′ defining the column, Y′ the row.

The discussion above for parameters applies particularly to X′ and Y′.

So the possible variations for coordinates X′, Y′ are

X+40B, Y+40B

where X and Y are in [0,136B]

COORDESCAPE, X1, X2, Y+40B

X+40B, COORDESCAPE, Y1, Y2

COORDESCAPE, X1, X2, COORDESCAPE, Y1, y2

Graphics Display Coordinates

The mouse is used to track the cursor on EITHER the Augment 1200 display or the Storage tube. A keyboard key acts as a toggle to select which screen is to be tracked. Graphics coordinates from the storage tube are sent as 10 bit values in the range 1024 to 2047, with 1024 at the lower left of the screen. They are sent as two bytes each of x and y.

Character Writing Position (cwp)

The terminal does not use a visible marker to indicate the character writing position. (an underbar is used for the separate function of tracking the mouse) However, such a position is always defined somwhere on the screen. Part of the screen (possibly all of it) is the TTY Simulation area. There is a TTY character writing position (ttypos) always defined within this area. The cwp may or may not be within this area.

The terminal has two basic states determined by whether or not cwp=ttypos:

Not Positioned:

Cwp=ttypos within TTY Simulation area. In this state characters received advance cwp and ttypos together.

Positioned:

Cwp#ttypos. Cwp operating as a result of a "position" command. Characters received in this state advance cwp, but ttypos remains unchanged. A "resume" command causes cwp to be set back to ttypos within the TTY Simulation area, and changes state back to "Not Positioned".

TTY Simulation

In TTY simulation (Not Positioned), scrolling always takes place on a line feed (LF) not a carriage return (CR). Carriage return does the obvious thing and no more.

Special and Control Characters

Protocol strings begin with 33B and are followed with an operation type character in the range 40B to 120B.

When outside a protocol string, all control characters (0 thru 37B) are treated as a space by the terminal, except:

0, which is ignored

^G which rings a bell

CR and LF which do the right thing when "Not Positioned".

When "Positioned", they are treated as a space.

^H which does a backspace position

33B which begins a Protocol string

(Also, the third mode button from the top on the front panel causes control characters otherwise treated as spaces to display special, identifiable characters.)

RUBOUT is ignored at all times, including anywhere within a protocol string.

Terminal to Main Computer Protocol

All keyboard characters (0B thru 177B) are sent out the line except:

The three control codes, 34B, 35B, and 36B are used for internal terminal functions. When they need to be sent, the fourth mode button on the display can be depressed, suspending their interpretations as terminal controls.

34B: (^\) Screen Save

35B: (^]) Graphics bug

36B: (^^) Screen Switch

Coordinate Mode (entered as a result of a command from the main computer)

In coordinate mode, 2B, 4B, and 30B are always preceded by:

↑B ↑D ↑X

34B, 42B, coordinates.

Also, in coordinate mode, 1B and 33B sent with the mouse buttons alone (100 and 101 binary) are preceded by

34B, 42B, coordinates.

For single button mouse characters (1B, 4B, and 30B) which cause cordinates to be sent, the mouse position when the button is depressed determines the coordinates sent, even though they are sent after the button is released.

Where multiple mouse buttons are involved in the sending of coordinates, it might be desirable to have the coordinates determined when the first of the buttons is depressed, but this is not how it is currently being done on the Augment 1200. Rather it is determined at the last mouse button transition to a non-zero state (generally when the next-to-last button is released).

Mouse and Keyset

Mouse/keyset interaction

The line traffic for mouse button changes is considerably reduced for the Augment 1200 as compared with the Line Processor.

Generally, changes in the mouse buttons are not sent except when nescessary, e.g. when sending viewspecs or markers with button positions 110, 111, and 101 binary.

Case shifting for the keyset (including control characters) is handled within the terminal. Characters sent with mouse buttons alone (such as OK) invoke the same line protocol as the cooresponding keyboard characters, except for 1B and 33B in coordinate mode, as described above.

Mouse button changes (in coordinate mode, when absolutely nescessary) are sent as:

34B, 43B, buttons+100B, coordinates

where buttons is the binary image of button positons (000 thru 111 binary).

Keyset strokes are sent as shown in (29988,)

BUTTONS:	000	010	100	001	110	011	101	111
KEYS:								
00000		<30B>	<01B>	<04B>	<27B>	<02B>	<33B>	<nothing>
00001	a	A	!	a		<01B>		
00010	b	B	"	b		<02B>		
00011	c	C	#	c		<03B>		
00100	d	D	$	etc.		etc.		
00101	e	E	%					
00110	f	F	&					
00111	g	G	'					
01000	h	H	(					
01001	i	I	)					
01010	j	J	@					
01011	k	K	+					
01100	l	L	-					
01101	m	M	*					
01110	n	N	/					
01111	o	O	^					
10000	p	P	0					
10001	q	Q	1					

10010	r	R	2
10011	s	S	3
10100	t	T	4
10101	u	U	5
10110	v	V	6
10111	w	W	7
11000	x	X	8
11001	y	Y	9
11010	z	Z	=
11011	,	(	[
11100	.	)	]
11101	;	:	_
11110	?	\	<33B>
11111	<40B>	<11B>	<15B>

There is no special interpretation of keyset (or keyboard) charcters for mouse buttons 001 binary (OK) since the OK function is used so much. This allows some overlap between a keyset stroke and a succeeding OK button from the mouse. So long as the keyset is released before the OK button is released, the keyset character will be sent before the OK.

The mouse/keyset discussions here refer to the current firmware supplied with the Augment 1200. An older scheme where all mouse button changes were sent (by the same protocol sequence described above) was used in the firmware of some very early versions of the terminal supplied by SRI. This scheme is also still used in down-line loaded terminal programs, particularly the ones for wide and long screen.

In graphics mode coordinates are X(bits 10 - 6 (MSB's)) + 40B, X(bits 5 - 0 (LSB')) + 40B, Y(bits 10 - 6) + 40B, Y(bits 5 - 0).

At power-up and after the "LOCAL RESET" button is pushed, the terminal signals the Main computer by sending:

(34B, 50B)

The purpose of this is to indicate to the applications program that the terminal is now in a "power-up" state with a blank screen. The applications program then typically recreates the screen.

Response to Interrogate Command

A response to the interrogate command is sent as a protocol string of this form:

34B, 46B, XMAX', YMAX', DTYPE, DTIM', F'

Where

XMAX is the maximum x coordinate

YMAX is the maximum y coordinate

DTYPE is in [40B-177B] and designates type

For compatibility with Line Processors:

The least significant four bits of DTYPE were used for line processors to designate display terminal type (call it DItype)

Those defined were:

(1) Delta Data 5200

(2) Hazeltine H2000

(3) Data Media Elite 2500

(4) Lear Siegler ADM-2

The most significant three bits designated Line Processor type (call it Type)

Defined:

(0) Complete alpha line processor with copy printer receiver for cassette drive

(2) Line Processor with Mouse, Keyset, Printer

(6) Graphics line processor with Tektronix 4014

(7) Graphics line processor with Tektronix 4012

For Augment 1200's, two types are currently defined:

DTYPE=45B: no Scroll Window capability

(virtually obsolete)

DTYPE=46B: Scroll Window protocol supported

DTIM' = Dtim + 40B.

Dtim is a characteristic delay time. Line processors required a line feed (LF) to be followed by (Dtim+14)/F pad characters. For the Augment 1200, Dtim is alwayz zero. In fact, a line feed requires no following padding.

F' indicates the Line Processor receive baud rate:

F' = F + 40B

where F = 9600/(receive baud rate)

Thus:

300 buad: F'= 100B, F=32 decimal

600 baud: F'= 60B, F=16

1200 baud: F'= 50B, F=8

2400 baud: F'= 44B, F=4

4800 baud: F'= 42B, F=2

9600 baud: F'= 41B, F=1

Note: Any additions to DTYPE should be assigned by TYMSHARE.

Printer Request for String

34B, 47B, DEVCODE, SEQNUM, CHARCNT'

Result:

Informs main computer that terminal is ready to receive a buffer of up to charcnt characters.

DEVCODE must be the same as received from main computer in a previous "Open Printer" command.

SEQNUM normally is in sequence from the last request except:

The terminal may rerequest due to receive error or timeout.

The terminal may skip up to four numbers over the previous SEQNUM which additionally implies a request for all of the skipped numbers with the charcnt.

CHARCNT' = max charcnt + 40B

Terminal sends charcnt=0 and SEQNUM = (seqnum of "EOF" indication) as a successful-completion-of-printing indication to the main computer.

See "Write Printer String" below.

From Main Computer to the Terminal

Padding

Some of the commands to the terminal listed below may require significant time (several character times) to complete. A 32-character input buffer relieves most timing restrictions, but a command that can take more than 32 character times (scroll window in particular) or a combination of such commands could cause the buffer to overflow if followed immediately by data or more commands.

The padding requirements listed below with the commands are worst case values. By following each command with the specified number of padding characters, there should be no buffer overfow even if an arbitrarily large number of such commands are sent in sequence.

In actual practice, only scroll window and clear screen require much attention.

Padding characters should be RUBOUTs (177B). The baud rate factor (F) and display type are obtained by the applications program by sending an interrogate command.

The following functions are sent by the applications program and performed by the terminal. All codes, except the escape (33B) should be printing characters.

Position cursor on display.

Send(33B, 40B, X', Y')

see discussion above on "Coordinates".

result:

Positions cwp (current writing position) to specified location. Any unescorted characters will be written on the screen and the cwp will be advanced once after each character. Writing beyond the end of the line wraps back to the beginning of the same line. If the terminal was "Not Positioned" before this command, the previous cwp is stored (in ttypos) until the next "Resume TTY" is received.

Specify (small) TTY simulation window

Send(33B, 41B, TOP', BOTTOM')

TOP' = Y' for top line of window

BOTTOM' = Y' for bottom line of window

result:

Invokes a TTY simulation window of specified size and location. This window will be used until a new one is specified or a reset is received. This does not change the position state.

Reset

Send(33B, 51B)

result:

screen cleared

TTY simulation window set to full screen

Not Positioned

padding:

40/F

Resume TTY window cwp (Not Positioned)

Send(33B, 42B)

result:

Any unescorted characters will go into the TTY simulation window currently defined. Cwp is restored (from ttypos) if the terminal is leaving the "Positioned" state.

Write string of blanks

Send(33B, 43B, N′)

N′: N = number of blanks to be written.

result:

The specified number of blanks are written starting at the cwp. Cwp is left at the character position following the last blank. Assumes the cwp has been Positioned appropriately beforehand.

padding:

This command must have (N/80)/F padding characters following it.

Bug selection

Send(33B, 46B, X′, Y′)

result:

The character at (X,Y) location is somehow brought to the user′s attention (reverse video). This command includes a resume TTY (Not Positioned).

Pop bug selection (unused on 1200)

Send(33B, 47B)

result:

This function was used on lineprocessors, but not on the Augment 1200. The applications program must restore the character at the bugged position to remove the bug indication.

Delete selected line

Send(33B, 44B)

result:

The cwp determines a line to be removed from the screen. All following lines are moved up one line. A blank line is added at the bottom of the screen.

padding:

This command requires 1/F padding characters.

Insert selected line

Send(33B, 45B)

result:

The line which the cwp is on, and all following lines, are moved down one line. The cwp is not moved, and hence is on a blank line. Lines above the cwp are not altered. The last line (before the execution of this command) is lost.

Padding:

This command requires 1/F padding characters.

Clear screen

Send(33B, 50B)

result:

The entire screen is cleared. The cwp is not generally known. The TTY simulation window location and the bug selection stack are not altered. The tracking mode is not changed.

Padding:

This command requires 28/F pad characters;

Interrogate

Send(33B, 55B)

result:

A response to the interrogate command is sent as a protocol string as described above under "Response to Interrogate

Command".

This command does not change the tracking mode.

Turn off coordinate mode

Send(33B, 60B)

result:

Turns off the coordinate mode in the Line Processor. This does not change "Positioned" mode.

Turn on coordinate mode

Send(33B, 61B)

result:

Turns on the coordinate mode in the Line Processor. This does not change "Positioned" mode.

Begin standout mode

Send(33B, 56B)

result:

All following text written on the screen will be in reverse video as compared to "normal" text. Does not change "Positioned" mode.

End standout mode

Send(33B, 57B)

result:

Subsequent text written on the screen will be in "normal" mode. Does not change "Positioned" mode.

Scroll Window

Send (33B, 65B, LEFTX', RIGHTX', TOPY', BOTY', N')

(in the current implementation on the augment 1200, the (unprimed) parameters are all limited to 0-255, but this is not a limitation of the protocol)

Scrolls arbitrary rectangular window up or down ABS(N) lines.

N > 0: scroll up

N lines lost at the top, N blank lines added at the bottom.

N < 0: (12 bit 2's complement) scroll down

ABS(N) lines lost at the bottom, blank lines added at the top.

Negative N always requires the extended (three-character) protocol.

Padding:

Every character in the area must be written in terminal memory. The command is slightly faster for bigger N.

The padding must be sufficient to give about 1 msec per 40 characters in the window plus 1 msec per 3 lines in the window.

Or, padding=((NC/40)+(NL/3))/F

Where NC = (TOPY-BOTY+1)*(RIGHTX-LEFTX+1)

And NL = (TOPY-BOTY+1)

Graphics, Printer, and Down-line Loading protocol

Graphics and Printer <characters> protocol:

In the following sections, <characters> refers to the some number (COUNT) of characters sent on the line to the terminal as part of a "Write Graphics Display" or "Write Printer String" command. 177B (rubout--line padding character) and 33B (escape) cannot be included in such a string. The following protocol is used to affect their being sent to a local printer or graphics terminal.

(26B, 101B, N')

sends N rubouts to the printer or graphics port. This accounts for three characters of "COUNT", though it generally causes some other number of characters to be sent to the printer or graphics port.

(26B, 102B)

sends an escape to the printer or graphics port.

(26B, CH) where CH is not 101B or 102B:

sends CH to the printer or graphics port.

In particular, (26B, 26B) sends 26B.

A general control character protocol is proposed as follows, though it isn't implemented yet.

(26B, 103B, CCHAR')

sends CCHAR to the printer or graphics port, where CCHAR' = CCHAR+40B.

Write graphics display

Send(33B, 52B, DEV, COUNT', <characters>)

Result:

The DEV is normally 40B and is ignored by the terminal. Characters from the application program are written directly on the graphics display. Since character buffering is limited, the graphics display should be connected at a higher baud rate than the external processor line.

COUNT' = COUNT+40B where COUNT <= 80. COUNT is the number of characters in <characters>.

Open Printer (or Open Down-line Load)

Send(33B, 64B, DEVCODE, MODECODE)

Printer: normally DEVCODE=40B

Down-line Load: DEVCODE=141B

Augment 1200 down-line loading of terminal programs also uses "Write Printer String" and "Printer Request for String" protocols. <characters> passed with "Write Printer String" are appropriate to produce "Down-line Load Format".

MODECODE is normally 40B.

If MODECODE .A 1 = 1 (e.g. MODECODE = 41B), this is an attempt to reopen printing and terminal will resume printing from its buffer and will make requests sequential to its last request.

Write Printer String (or Down-line Load String)

Send(33B, 63B, CHKSUM2', CHKSUM1', DEVCODE, SEQNUM, COUNT', <characters>)

checksum: (integer addition starting with DEVCODE):

CHKSUM2 = (checksum low 4 bits) + 40B

CHKSUM1 = (checksum high 4 bits) + 40B

DEVCODE: same code as passed with "Open Printer".

SEQNUM: starts at 40B, runs sequentially to 175B, the starts over.

COUNT' = (number of characters in <characters>) + 40B.

COUNT' = 40B (COUNT=0) indicates EOF to the terminal.

Terminal will re-request on checksum error or after non-receipt of requested string for about 25 seconds. (see "Printer Request for String" under Terminal to EP protocol)

Close Printer

Send (33B, 54B)

This command would be better described as "Abort Printing". It is not normally used, as may be inferred from the description of EOF indication under "Write Printer String". The Close Printer command causes immediates cessation, discarding any contents of the printer buffer in the terminal. Printing cannot be "reopened" after this command (i.e. with the "reopen" mode of the "Open" command).

Down-line Loading

General

There are two modes to send terminal programs down line. One has been alluded to above, with the down-line load protocol as the data part of the same protocol as used for the printer. The other uses the down-line protocol by itself as a strictly main-computer-to-terminal protocol, which is possible since this protocol has the same form and does not conflict with the commands described thus far. The only change nescessary for using it in the printer-like mode is to convert each escape (33B) to a two-character (26B, 102B) sequence.

Comparison of the two modes:

The printer-protocol mode is usually preferred since the data is checksummed. The "raw" mode is slightly faster since the "escape"s are sent as single characters, but it requires perfect transmission from the main computer to the terminal. The Printer-protocol, however, requires quite reliable transmission from terminal to main-computer, so it often can't be used on ARPANET TIPs with small input buffers.

The protocol is appropriate for an eight-bit device (such as the 8080) where addresses are 16 bits or less.

Down-line Load Protocol

Set Address

Send(33B, 100B, AP0, AP1, AP2, AP3)

AP0 = A0 + 100B

AP1 = A1 + 100B

AP2 = A2 + 100B

AP3 = A3 + 100B

A1 IN [0, 17B]

ADDRESS = A3*10000B + A2*400B + A1*20B + A0

ADDRESS is a 16-bit quantity

Sets the memory address for a subsequent "Memory Data" command.

Memory Data

Send (33B, 101B, DP00, DP01, DP10, DP11, ...)

DPi0 = Di0 + 100B

DPi1 = Di1 + 100B

Dij IN [0, 17B]

Di = Di1*20B + Di0

Di is the 8-bit data for the i-th memory location after "ADDRESS" set by "Set Address" command.

Data is loaded into sequential memory locations until an "End Memory Data" is received.

A pair of charcters is sent for each location to be loaded. A "Set Memory" command is typically 66 characters long and thus sets 32 locations.

End Memory Data

Send (33B, 102B)

Terminates "Memory Data" string.

Application notes:

Avoid writing text (or "string of blanks") beyond the end of a line: the display currently wraps to the beginning of the same line, but this feature isn't guaranteed.

Avoid positioning the cursor to any x>Xmax or y>Ymax.

NOTE:

The terminal has a hardware reset button on it. After power up or a hardware reset, the following state prevails:

The screen is clear, "Not Positioned".

The full screen TTY simulation is in effect.

Cwp is at upper left corner of screen

Coordinate mode is NOT in effect.

Printer is closed

All TTY simulation windows currently work as follows: When cwp reaches the last line, "scrolling" occurs on each line feed. A CR moves the cwp to left margin, a LF effects a line break. Typing beyond the last character of the line causes a line "wrap" - i.e. new text replaces the old line, starting from the left margin. The way to clear a small TTY window is to send N line feeds into it, where N is the number of lines in the window.

The Scroll Window command may be used to clear arbitrary windows.

In manipulating the display, the usual sequence from the applications program will be to position the cursor and perform some function, or write text, or both. It must end such a sequence with a "resume TTY" command. Any broadcast messages, links, etc. that come down the line between the "position" and the "resume TTY" will go wherever the cwp happens to be.

Normally, broadcast messages and the like will go into the TTY simulation window since the terminal spends most of the time in the "not positioned" state. Of course, such messages are not preceded by a "Position" command.

REENTER code in Augment will clear and repaint the entire screen

A tracking mark (underline) on the screen tracks the mouse except while graphics tracking is in effect.

Summaries

Terminal to Main Computer

CHAR SEQUENCE MEANING

CHARACTER Normal Character

Ascii values 1B to 177B except:

when in coordinate mode:

from keyboard, keyset, or mouse:

2B (^B), 4B (^D), 30B (^X), and 34B (BCESC)

from mouse only:

1B (^A), 33B (esc)

34 46 MX' MY' TP DT' F' Interrogate Response

34 43 MB X' Y'   Sequence For Mouse Buttons

34 45 MB X1' X2' Y1' Y2'   Sequence For Mouse Buttons
(when graphics tracking)

34 47 DV SQ CNT'   Request for Printer String

Where:

All numbers are in octal

Prime (') indicates 1 or 3-character parameter sequence as described in "Protocol Parameters".

MB = current mouse button state + 100

X' = current x coordinate (+ 40 or 3-char sequence)

Y' = current y coordinate (+ 40, etc.)

X1 = top 6 significant bits of x coordinate + 40

X2 = least significant 6 bits of x coordinate + 40

Y1 = top 6 significant bits of y coordinate + 40

Y2 = least significant 6 bits of y coordinate + 40

MX' = maximum x coordinate (+ 40, etc.)

MY' = maximum y coordinate (+ 40, etc.)

TP = terminal type and version

DT' = terminal delay time characteristic (+ 40, etc.)

F' = 9600/(line receive baud rate) (+ 40, etc.)

DV = printer device code

SQ = sequence number

CNT' = max chars request (+ 40, etc.)

Main Computer to Terminal

COMMAND	CODE	PADDING

position	33B, 40B, X', Y'	none
TTY window	33B, ! 41B, YTOP', YBOTTOM'	none
resume tracking	33B, " 42B	none
write blanks	33B, # 43B, N'	(N/80)/F
delete line	33B, $ 44B	1/F
insert line	33B, % 45B	1/F
push bug	33B, & 46B, X', Y'	none
clear screen	33B, (50B	28/F
reset	33B,) 51B	40/F
graphics string	33B, * 52B, DV, CNT', String	see text
close printer	33B, , 54B	none
interrogate	33B, - 55B	none
standout mode on	33B, . 56B	none
standout mode off	33B, / 57B	none
coordinate mode off	33B, 0 60B	none
coordinate mode on	33B, 1 61B	none
open printer	33B, 4 64B, DV, MODE	none
printer string 33B, 63B, CKS2', CKS1', DV, SQ, CNT', <characters>		none
scroll window 33B, 65B, LEFTX', RIGHTX', TOPY', BOTY', N'		((NC/40)+(NL/3))/F

NC = (RIGHTX-LEFTX+1)*(TOPY-BOTY+1)

NL = (TOPY-BOTY+1)

```
OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OF
OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OF
OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OFFICE-2   OF


LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN
LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN
LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN   LEHTMAN


$HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $H
$HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $H
$HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $HGL$75166   $H


Monday, August 18, 1980 10:16:56-PDT   Monday, August 18, 1980 10:16:56-PDT   Mo
Monday, August 18, 1980 10:16:56-PDT   Monday, August 18, 1980 10:16:56-PDT   Mo
Monday, August 18, 1980 10:16:56-PDT   Monday, August 18, 1980 10:16:56-PDT   Mo
```

```
< AJOURNAL, 75166.AUG;1, >, 11-Aug-80 17:31 XXX ;;;;   Title:
Author(s): Thomas R. Davis/THOM; Distribution: /DIA( [ ACTION ] ) GTK( [
ACTION ] ) DLS( [ ACTION ] ) DAP( [ ACTION ] ) PKA( [ ACTION ] ) JCN( [
ACTION ] ) HGL( [ INFO-ONLY ] ) AMP3( [ INFO-ONLY ] ) MEB3( [ INFO-ONLY
] ) KIRK( [ INFO-ONLY ] ) CLEN( [ INFO-ONLY ] ) JDH( [ INFO-ONLY ] )
SGK( [ INFO-ONLY ] ) ; Sub-Collections:  NIC; Clerk: THOM;
Origin: < DAVIS, IDENT-COMMENTS.AUG;1, >, 11-Aug-80 16:02 THOM
;;;;####;
```

THOM 11-Aug-80 16:05 75166

Comments on proposed ident system design

Responds to <49476,>

Comments on proposed ident system design

Following is a short collection of comments on DIA's and GTK's <49476,> -- Draft of New Identsystem Design. The comments are listed in approximately the same order as the things they refer to are listed in the journal document. 1

One question I had a number of times while reading through the document is: If multiple ident systems are to be supported, who will keep track of all the names (of ident systems), or does anyone really need to? Obviously, some related systems will need to know about each other, but what will be the policy for ident systems on hosts not connected to any networks? Shall we require all creators of new ident systems to apply to us for a "blessed" name and number? It seems to me that this is essential, if we really want to be able to figure out exactly who every ident is. If we keep this information, where and how is it kept? 2

Another somewhat related question is this: Are any of our users out there going to have access to the source code? What if they change the code in their ident systems? Do we want to specify the various protocols for interrogation of other ident systems at least? 3

Do we want to specify exactly what information is to be associated with each ident? Name, hard-copy address, on-line address, ... , what else? Will there be a facility for adding new types of information later, should it become important? We might now want to list something about network access privileges, for example. 4

On making changes to the ident database, I assume that all changes are made on a single master host, and then the files are automatically transferred to the other hosts for the ident system. Is that right? How many files will need to be transferred? How long will it take, especially over tymnet? What kind of protection will there be to keep people out of the ident system when only half of the new files are across? All of these things can be solved, but some may be messy. 5

Concerning ident system protections, is the privacy on individual idents, or on whole ident systems? Will every ident system automatically have a public "in" box? It seems to me that it would be easier to always allow one to send mail to any ident if it is known. The ident database may be secret, but if I know the name XYZ@SECRET, I should be able to mail a message to him -- sort of like the US mail system. We should certainly make it possible to restrict who has the right to send mail to be recorded in a particular journal, but maybe I, as a member of ident system A, should be able to send a message to XYZ at B, having the message recorded only in my journal, with a citation (and unrecorded copy) sent to XYZ. 6

What is the relation between ident systems and journals? One ident system per journal, or what? I did not see any discussion of that in the design document. 7

How difficult is it going to be to add the information about the ident system involved to the statement signature information? Are we going to add a new property to each statement in the souped-up AUGMENT? I doubt that there is room for more information in the present statement signature. 8

What is meant by ident systems being potentially as large as reasonably possible? Does this mean 10000, 25000 or 100000? 25000 might be a good size which would allow about 10 words per entry in the index files. I looked in the ARPA directory and there were about 3000 names. In the old NLS ident file before I culled out the "expired" idents, there were about 2500. Maybe 10000 would be plenty -- surely enough if we only consider active idents. 9

I have a number of questions about the actual structures of the idents. Exactly what is the difference between a Group and an Organization? Will groups be allowed to be members of groups, of organizations? How about vice-versa? How deeply can this be nested? If we allow nesting, how do we prevent circular definitions, or do we want to? Deletion of group idents can be very messy when things are members of other things, and we delete something out of the middle of such a chain. Will it be possible to define group idents which consist of members of different ident systems? If not, we miss out on some interesting possiblities, but if so, circularity checks would take practically forever. 10

How often will the various hashed quick look up files be updated? With each ident insertion? Will the whole things be regenerated, or will they be edited by an appropriate program. With all these index files to update, it seems to me that the process of adding an ident could be much slower than it is now. With all of the separate lookup files, we must provide a mechanism for regenerating all of them at once from the source file in case they get clobbered. 11

How do we keep the source file from getting clobbered? We had better have some good verify command that will spot trouble early. 12

I didn't understand exactly what the source file was. Is it just an AUGMENT editable file, or will it be editable only via the ident subsystem? I think that there had better be some way to edit it via AUGMENT (maybe only by someone with special capabilities), because no matter how careful we are, something is bound to go wrong with it. I read some things about information being stored in properties of statements -- was this the planned source file or what? 13

If the ident system is going to get large enough to require more than one file to contain all of the source information, how will the information be divided among the various partial files? It would probably be convenient to have them sorted in some way, with A-L in file one and M-Z in file two or something like that. This would make the ident files useful for generating such things as mailing lists and on-line directories a la JAKE. If the files are sorted this way, however, cross-file sorting may prove to be a problem to do efficiently. 14

How hard will it be to add a new ident system? Who will do things like this? 15

350.33
821 60
1171 93
319 00
852 93

FOONLY

SYSTEM XXV

FAILURE RECOVERY PROCEDURES

March, 1981 DRAFT
Version 2
Journal Number 75400

Office Automation Division
Tymshare, Inc.
20705 Valley Green Drive
Cupertino, California 95014

TABLE OF CONTENTS

1 INTRODUCTION

1.1 The System XXV

The brain of the System XXV is a Foonly computer and its memory. Associated with this are three disk drives, which store and retrieve information on the disks, and a tape drive for reading information into the system from tape. In addition, the system is connected to one or more networks which allow it to communicate with users and other computers. These various parts which make up a System XXV are run by a monitor (or operating system) called "AUGUST". AUGUST talks to the network, makes sure that all the parts of the system are working properly and in harmony, and oversees users' interaction with the programs run by the system. AUGUST also communicates directly with System XXV users through a program called "EXEC" and carries out many user commands.

1.2 Purpose and Structure of the Manual

This manual provides the information necessary to bring up a System XXV that has crashed. It is divided into several major sections, each of which covers a different situation you might encounter after a system crash. Their order is the same as the series of questions you might ask yourself when faced with a system that is not operating correctly. We hope that reading through the sections in order will enable you to step through the process of determining what is wrong with a system, deciding how to bring it up, and finally actually doing the recovery procedure you have decided on. Because the manual is arranged in this working order, its first several sections deal with error conditions and hung systems. Only after these problems are dealt with can we turn in section 5 to what to do if the system has crashed. This section discusses recovery procedures in general, the various types of recovery available on the System XXV, and when to use each of them. Section 5 is very important: do not skip it.

Where applicable and helpful, each large section of this manual is divided into four parts:

Introduction. A quick look at the current section. This will tell you such things as what the section is about, how the information is organized, and where to go if it is not what you need.

Summary. An outline that briefly presents exactly what you need to know or do. This is meant to be used as a working document or for quick reference; no explanation is included.

Discussion. A detailed explanation of the information and procedures outlined in the Summary.

Errors and Recoveries. A list of common problems and suggested solutions.

1.3 Conventions of the Manual

This manual has a set of conventions that will make it, we hope, clear and easy to read.

Program names always appear in capital letters, for example, CHECKDISK.

Special keys on the terminal are indicated by the abbreviations:

<SP> for space

<CR> for carriage return

<ESC> for altmode or escape

<LF> for line feed

Control characters are indicated with the notation "CTRL" and surrounded by angle brackets, for example, <CTRL-X>. To type a control character, hold down the CTRL key while typing the letter. To type <CTRL-X>, for instance, you would hold down the control key and at the same time type an X (in uppercase or lowercase).

The manual will refer to switches on the control panel by function in capital letters. When you look at the control panel, you will see that the switches are in rows and that different rows of switches are labeled by what they control; for example, there is a row of switches labeled "micro processor". Inside these rows of switches, individual switches are named by what they do: for instance, in the row of switches labeled "micro processor", there is a switch named "stop". This switch, which is used to stop the microprocessor, is called MICRO PROCESSOR STOP.

You put control panel switches "on" by pushing them up, and you put them "off" by pushing them down. Some switches are momentary, which means that after you put them on (up), they will return to the off (down) position when you release them. If you read "Put MICRO PROCESSOR STOP on", this means push up the switch labeled "stop" in the row of switches labeled "micro processor".

Commands appear in two ways. When the command is discussed in the text, the first letter of the command is capitalized and there are no quotation marks. When, on the other hand, you are directed to enter a specific command, for example in the Summary of a section, the command and its argument(s) are lowercase and enclosed in quotation marks. In this case, type exactly what you see, excluding, of course, the quotation marks. If the operator's terminal is uppercase only, you may type the commands in uppercase: however, the reverse is not true. All uppercase commands must be given that way; do not type them lowercase. Some prgrams cannot recognize uppercase letters.

2 ERROR CONDITIONS -- HOW TO IDENTIFY THEM

2.1 Introduction

When you are faced with a system experiencing some problem, the first thing you need to do is determine what this problem may be. There are three important aids in this process: error messages, BUGHLT numbers, and error lights. Always read and record the error messages, BUGHLT numbers, and error lights before you attempt to bring the system up.

2.2 Error Messages and BUGHLT Numbers

When the system crashes, it usually provides an error message on the operator's terminal specifying what caused the crash. This is followed by a BUGHLT number. Always read and record error messages and BUGHLT numbers when you have a system that is down. A list of the various BUGHLT numbers and what they mean comes with this manaul.

The system may be set so that it does not print the BUGHLT number, but only prints the word "BUGHLT" followed by the location of the BUGHLT. When this happens, type ".[" to force the system to print the number. The BUGHLT number will be the second, or right, half of the number printed.

2.3 Error Lights

When the system is functioning normally, certain lights on the control panel are on, others off. When the system crashes, these lights change. Lights that indicate the system is functioning correctly are replaced by error lights, lights indicating some error has occurred. This section will help you tell the difference between lights lit during normal operation and error lights.

Normal Lights

When the system is operating normally, a pattern consisting of four lights will be cycling among the address lights on the control panel. Unless the system is very heavily loaded, these lights should be moving. If they do not move for a reasonable period of time, the system is probably hung or down.

Error Lights

The following lights on the control panel are lit steadily only when an error has occurred and the system has crashed or will crash.

MEM PAR ERR light indicates a memory parity error.

MI PAR ERR light indicates a microcode parity error.

PROG HALT light indicates that the computer has encountered a halt instruction in AUGUST, the operating system. Systems programmers occasionally install halt instructions in AUGUST to help them trace problems.

3 IS THE SYSTEM HUNG OR HAS IT CRASHED?

Before you can deal with a system that is not operating correctly, you must determine whether it is hung or has crashed. Learning to recognize a hung system is a matter of practice. There is no one sure test that will determine if a system is hung, but hung systems do have the following common symptoms.

1) Lights on the control panel appear static or are immobile and pulsing in some kind of regular pattern.

2) There is no response when you type <CTRL-C> or <CTRL-T> on the operator's terminal.

3) You cannot log in from another terminal.

4) You are receiving irate calls from users who are unable to do anything.

5) In spite of all this, there is no BUGHLT indicated on the operator's terminal.

If a system is not functioning and does not have one or more of these symptoms, then it has crashed. See section 5, What to do if the System has Crashed.

4 WHAT TO DO IF THE SYSTEM IS HUNG

4.1 Introduction

The procedure documented in this section will force a hung system to crash. This may seem brutal, but it is necessary. The hung system is in limbo; only after it crashes can you bring it back up. When you finish this procedure and the system is down, use the disk recovery procedure to bring it up.

4.2 Summary

1) Put address switch 31 on (up).

2) Put data switch 2 on.

3) Put CONSOLE DEPOSIT THIS momentarily on.

4) Put data switch 2 off.

5) Put data switch 0 on.

6) Put CONSOLE DEPOSIT THIS momentarily on.

7) Wait until activity (the flickering of the lights, etc.) stops.

8) Bring the system up with the disk recovery procedure.

4.3 Discussion

When the system is hung, it is trapped in the execution of some process. The procedure outlined in the above Summary is designed to bring the system out of this cycle and cause it to crash. This is desirable because crashing is the normal response to abnormal conditions. When it crashes, the system tries to take care of itself -- to save files, to protect the monitor, to print an error message indicating what the problem may be, and so forth. Furthermore, only after it has crashed can the system be brought up.

Because a hung system ignores commands entered on the operator's terminal, to work with one, you must enter the data and commands manually from the control panel. Put on address switch 31 by pushing the switch up. Then, put on data switch 2. Finally, momentarily put on CONSOLE DEPOSIT THIS. This process turns on bit 2 at address 20 octal in the computer's memory. When you turn on this bit, you tell the system that everything that is stored in the temporary storage area should be read back into its permanent location. Temporary storage contains all new information the system has not read out to its real disk location and also the intermediate results from processes being performed but not yet completed. Before forcing the system to crash, you need to make sure that all this information is safely stored in the right place on the disk.

Now put data switch 2 off, put on data switch 0 and then put on CONSOLE DEPOSIT THIS. By doing this, you turn on bit 0 at address 20 octal. This bit is used by the system to record and check its status. When bit 0 is off, the system knows it is running successfully; when bit 0 is on, it means the system has encountered a dangerous situation and should crash. Thus, when you turn on bit 0 manually from the control panel, you trigger a system crash. Once all the flickering of the lights stops, the system is down. Bring it up with the disk recovery procedure.

5 WHAT TO DO IF THE SYSTEM HAS CRASHED

5.1 Introduction

When a System XXV crashes, before you can bring it up you must decide which recovery procedure to use. This section will help you do this. It is divided into three parts. The first part explains what the System XXV's recovery prodecures do, the second part briefly describes the recovery procedures available, and the third part will help you decide which procedure you need to use. In addition, we give advice about what to do if you cannot bring up the system. Once you know which recovery procedure to use, for specific instructions, go to the major section describing it.

5.2 What is a Recovery Procedure?

The System XXV is operated by a very large program called the "monitor". The monitor is basically what makes a machine into a computer. It is responsible for checking the system to make sure it is running correctly, transferring information within the computer and between it and the outside world, overseeing all the various programs run by the users, keeping the users' jobs separate and allocating resources to them, and so forth.

In keeping with its two functions, running the system and overseeing the users' programs and requests, the monitor is divided into two parts. The most important part is the "resident", or "kernel", monitor. The two names of this part of the monitor reflect its two major characteristics. "Kernel" monitor indicates that this part of the monitor is the core of the system. It contains basic instructions and information necessary for the system to function. For this reason, it must always remain, or reside, in central memory; thus, the name "resident" monitor.

The second part of the monitor, the "swappable" monitor, contains information and procedures related to users' needs rather than system functions. It is called the "swappable" because, unlike the resident monitor, this part of the monitor is not always present in central memory. Instead its various parts are copied or "swapped" into central memory only when they are needed. The Copy File to File process is an example of the type of procedure located in swappable monitor. When you enter a Copy command, the system begins by looking for this process in the parts of swappable monitor present in core. If it discovers that the Copy process is no longer im memory, it recalls it from disk and then executes your command.

The System XXV's entire monitor is called "AUGUST". AUGUST thinks it is running on a PDP10. Since it is not, AUGUST depends on another part of the system called the "microprocessor". The microprocessor is what translates the monitor's instructions into something the System XXV can understand. It consists of a memory containing information called "microcode", and a "microcontroller" that uses this information. When AUGUST, the monitor, gives a PDP10-like instruction, the microcontroller takes the instruction and uses the microcode to translate it into the equivalent instruction for a System XXV.

In most System XXV crashes the problem is an error in the monitor. The System XXV's various recovery procedures are designed to replace the old copy of the monitor with a new one from disk or tape where copies are permanently stored. Some crashes, however, destroy not only the monitor but also the microcode. Since the microprocessor cannot function without the microcode, this means that the microprocessor can no longer translate the instructions you or the monitor try to give it. In this case, before copying the monitor, the recovery procedure must also provide a new copy of the microcode. After the microprocessor has this copy of the microcode, the system is given the resident monitor. Once the resident monitor is safely stored in central memory, the system starts running and then copies the swappable monitor.

After the new monitor is in memory, in most recovery procedures, the system checks the file system with a program called CHECKDISK. If everything is OK, the system reports "August in operation". This means the system is ready to come up and open itself for normal use. If CHECKDISK discovers something wrong with file system, it will not come up. Instead, it will wait for you to correct the problem. After correcting the problem, you will have to halt the system and bring it up again. (As you will learn in the next section, some recovery procedures allow you to avoid this system checking.)

It should be emphasized that the reocvery procedures are simply programs like the monitor and everything else that runs on the system. They will not fix any hardward problems, and, in fact, cannot work if the system has something physically wrong with it. If you suspect that the system has a hardware problem, or you cannot bring it up after trying repeatedly, you may need to contact your manager and Tymshare maintenance.

5.3 Recovery Procedures Available

Introduction

The System XXV has five recovery procedures. None of them is particularly difficult, but they do have substantial differences. They are divided into two groups: those procedures which return the system to normal use, and those procedures which should be used only after very serious system error and which do not return the system to normal use. The Summary below lists all five procedures and mentions one or two of their most important features. Following this is a general discussion of what each procedure does and of the procedures' relationship to each other. To decide which procedure to use after a crash, see the next section. For details on exactly how each procedure works, see the individual section which discusses it. To learn about recovery procedures in general, see section 3.2, What is a Recovery Procedure.

Summary

Each of the following three procedures returns the system to normal operation and opens it to users.

Disk Recovery. You instruct the system to look on the disk for the information it needs to come up. Discussed in section 6.

Tape Recovery. You provide the information the system needs to come up from tape: providing a new copy of the microcode is an optional part of this procedure. Discussed in section 8.

Automatic Recovery. After crashing, the system immediately copies the information it needs from disk and tries to bring itself up. This is done automatically, without waiting for an operator. Discussed in section 7.

Both the next two procedures brings the system up closed to normal users and allows systems programmers to investigate what is going on. Use them only as a last resort, after serious system errors, and under supervision

Standalone Recovery. You instruct the system to come up without checking the file system or running the system jobs. Discussed in section 9.

Disk Rebuild. Before bringing the system up, you wipe out and then rebuild the entire file system, reading copies of every file from tape. Discussed in section 10.

Discussion

When the System XXV crashes, in most cases you bring it up by replacing the old copy of the monitor with a new one. You can provide this new copy either by copying it from disk, in which case you are doing a "disk recovery", or by reading it from tape for a "tape recovery". Both of these procedures are begun by an operator after the system has crashed. As part of the tape recovery procedure you may also read in new copy of the microcode the information used by the microprocessor. Replacing the microcode is usually necessary only after crashes due to power failure.

In addition to the tape and disk recovery procedures, there is another procedure that the system itself can start up after a crash. Since the system begins this procedure without waiting for anyone to instruct it, this third type of recovery is called "automatic recovery". Automatic recovery is very much like disk recovery. Upon crashing, the system immediately copies the current contents of central memory to disk, copies in a new monitor from disk, and starts to bring itself up. Automatic recovery never occurs unless the system was already set for it before crashing. To learn how to set a system to recover automatically, see section 12, Recovery Switches.

Disk, tape, and automatic recovery have substantial differences -- they are either automatic or not and the new monitor comes from either tape or disk. However, all three have the same result: they all end with the system checking itself and the file system and then being opened for normal use. The next two recovery procedures do not have this convenient result.

Both standalone recovery and disk rebuild allow the system to skip important parts of the normal recovery procedure. For this reason they are very risky; do not attempt them without being specifically instructed to do so and without supervision of a systems programmer or manager. In standalone recovery, the monitor is read from tape and you then direct the system to bypass its normal self-checking procedures and come up CLOSED to users. This means that only the operator's terminal has access to the system; no other users may log in. When a system will accept imput only from the operator's terminal, it is said to be "standalone". The standalone recovery procedures takes its name because it has this effect.

Even more serious than standalone recovery is disk rebuild. Disk rebuild allows you to do just what you might suspect from its name -- rebuild the file system stored on the disk. As in standalone recovery, you begin by reading in a tape containing a new monitor. Then, before the system needs to use any information from the disk, you begin the disk rebuild procedure. A disk rebuild involves destroying all the current versions of every file, and returning to the version stored on dump tapes; normally, it should NEVER be used.

5.4 Deciding Which Recovery Procedure to Use

Introduction

When a System XXV is down, whether for the first time in a month or minutes after a previous crash, the first step in bringing it up is deciding which recovery procedure to use. This section will help you make this choice. It is divided into two parts. The Summary contains a table showing types of crashes and their recovery procedures. The Discussion explains the logic behind the table; it will tell you why a particular recovery procedure is used in a certain set of circumstances. Once you determine which recovery procedure you need, go to the section discussing it to learn how to use it.

Summary

The table below shows when to use each of the System XXV's five recovery procedures. The left column lists different situations you might encounter; the right column shows the recovery procedure you should use. Note that you cannot decide to use automatic recovery after the crash has occurred. (Automatic recovery means that the system will try to bring itself up after a crash without waiting for an operator.) For automatic recovery to occur, the system must be set for it before the crash.

Situation	Recovery Procedure
Crash NOT DUE to power failure	Disk Recovery
Hung system is forced to crash	Disk Recovery
Any "normal" crash	Disk Recovery
After CHECKDISK problems corrected	Disk Recovery
Recovery begins automatically	Automatic Recovery
Crash DUE to power failure	Tape Recovery
Disk recovery fails	Tape Recovery
Automatic recovery fails	Tape Recovery
Tape recovery fails repeatedly	Standalone Recovery FIRST. contact manager or systems programmmer
Entire file system destroyed	Disk Rebuild FIRST, contact manager or systems programmmer

Discussion

The System XXV has three standard recovery procedures: disk recovery, tape recovery, and automatic recovery. In addition to these, there are two more risky recovery procedures, standalone recovery and disk rebuild, which should not be used without your manager's approval. This large number of choices means that you have more flexibility in responding to a crash, but it also means that you have more choices to make. Before you can bring up a system that is down, you must decide which recovery procedure to use. To make this decision, you must consider: 1) how the system will respond when it encounters an error while running; 2) the circumstances of the crash, what caused it, and what effect it had.

When a System XXV crashes, the first thing it does is check four internal switches, called "recovery switches". Recovery switches are four memory locations whose values tell the system how to respond to a crash. Recovery switches explained in section 12; here it is enough to know that they will tell the system to do one of two things: stop and wait for an operator, or immediately begin the automatic recovery procedure and try to come up. You should know how the recovery switches of each system have been set so you will know how it will respond to a crash. If you see a system crash and do not know what it will do, watch the system, until you know whether it is going to come up automatically or you need to begin a recovery procedure. If you see a system crash which you know is set for automatic recovery, it is wise to keep an eye on it and make sure it really does begin the automatic recovery procedure. Recovery switches are occasionally destroyed in system crashes. When this happens, a system originally set to recover automatically will simply sit there waiting.

If a system does manage to begin an automatic recovery, you have at first no decisions to make. If all goes well, the system will come back up and you will not need to do anything. However, automatic recovery does have two pitfalls. First, the recovery may not be successful and the system may hang or crash again. If you notice this happening, do not let another automatic recovery begin; the procedure is hardly likely to succeed on a second try. Instead, halt the system, if necessary, and bring it up yourself with the backup recovery procedure, tape recovery. The second problem that can keep the system from coming all the way up is errors in the file system. As the system comes up, it uses a program called "CHECKDISK" to examine the disk and make sure the files are OK. If CHECKDISK discovers problems, the system will stop to wait for someone to correct them. After correcting the errors, you will have to halt the system, and bring it up with disk recovery.

If, after a crash, a system does not try to come up automatically but instead just sits there waiting, then you must take over and begin some recovery procedure. Your choices at this point are disk recovery and tape recovery. Of these, disk recovery is more convenient and should be tried first, simply because it is so easy to use. However, disk recovery requires that the procedure itself survive the crash and that EDDT be available to begin it. Both of these are part of the old monitor. Crashes due to power failure destroy the old monitor (and microcode) completely and thus wipe them out. If you think the crash was the result of power failure, do not use disk recovery; instead, try tape recovery.

If you do decide to start with disk recovery and the system succeeds in running CHECKDISK, this means EDDT and the recovery procedure itself are OK. Even if CHECKDISK discovers file problems and the system does not come up, you may again use disk recovery, after correcting the problems and halting the system. File problems do not indicate that anything is wrong with the actual recovery procedure. However, if the first time you try disk recovery the system does not get as far as running CHECKDISK -- the recovery procedure never starts or the system crashes or hangs -- you will know that either EDDT or the disk recovery procedure or both did not survive the crash. In these circumstances, it is a waste of time to try the procedure a second time. Instead, halt the system, if necessary, and switch to tape recovery.

Tape recovery, the last of the three "normal" recovery procedures, is the backup procedure. In tape recovery the system copies the information it needs from tape; thus, recovery does not depend on any part of the system being able to function. Instead you enter all commands to the system through the control panel. However, although tape recovery is the most reliable of the recovery procedures, it is also the most time-consuming and inconvenient. Do not try tape recovery if you think disk recovery will work.

If you must use tape recovery, because the power failed or disk and automatic recovery do not work, feel free to try it several times. If you are using it after a power failure, or if nothing happens when you try to read the monitor tape, begin the procedure by reading in the microcode tape. If recovery does start, but the system never reaches the point of running CHECKDISK, halt the system, if necessary, and try the procedure over, again beginning by reading in the microcode tape. If, after trying the recovery three times from beginning, the system still does not run CHECKDISK, something may be seriously wrong. Notify a systems programmer or your manager; they may want to try standalone recovery. Once CHECKDISK does run, even if CHECKDISK discovers problems with the files system, the new monitor is in memory and this part of the recovery procedure has been a success. In addition, since EDDT and the disk recovery procedure are part of the monitor, they are again available. If CHECKDISK detects file problems and you must halt the system after correcting them, you may use disk recovery to bring the system back up.

Standalone recovery is the procedure used when some problem with the file system, CHECKDISK, the system jobs, and so forth, is causing the system to crash after the monitor is read into memory, but before it can come up all the way and return to normal use. In standalone recovery, after the monitor is copied from tape, the system simply stops where it is and waits. A systems programmer can then examine the monitor, the file system, and so forth. and try to determine what is wrong. A standalone recovery is not hard to do, but since the system comes up without checking how it is operating or making sure the file system is good, great harm may be done by mistake. Never undertake standalone recovery without expert supervision.

The final type of recovery procedure, disk rebuild, should be used only after a systems programmer has determined that a crash has damaged the file system beyond all hope of repair. Disk rebuild allows you to bring up the system in such a way that before anything is needed from the disk, the entire file system is replaced with backup files from tape. Because files can be replaced only with their most recent backups, the most current versions of many files will be permanently lost. The decision to do a disk rebuild can be made only by a manager or a systems programmer, and we hope the procedure will never have to be used.

5.5 Difficulty Bringing Up the System

If you cannot bring up the system or feel that something mysterious is going on, call Tymshare Maintenance or an OAD operating systems programmer.

6 DISK RECOVERY

6.1 Introduction

You should try to bring the System XXV up with the disk recovery procedure after any crash that is not due to power failure. The procedure is quick and easy; however, it relies on part of the monitor surviving the crash. This means it may not always work. Do not try to use disk recovery more than ONCE. If your first attempt at bringing up the system with disk recovery fails before CHECKDISK is run, you must halt the system, if necessary, and then switch to tape recovery. If CHECKDISK does run, then the system's new monitor is in place and this part of recovery has been successful. Even if CHECKDISK finds problems with the file system and you must halt the system after taking care of them, you may again use disk recovery to bring the system back up. To correct problems found by CHECKDISK and to halt the system, see section 13, Related Procedures.

6.2 Summary

1) Check the BUGHLT number on the operator's terminal and look it up in the list of BUGHLTs; in addition, note which error lights are lit. Record all this information.

2) On the operator's terminal, type "dskrld<ESC>g".

3) The reponse should be "reloading from disk". If you never get this message, begin a tape recovery.

4) When the system says, "BOOT FROM DISK PACK # [CR FOR ANY]", type <CR>. The system will begin to copy the monitor and will record its progress in messages.

5) When the operator's terminal says "EDDT", type "start<ESC>g". After a short time, the system should report the size of the memory and print several messages about BAT blocks.

7) CHECKDISK will run and check the file system. If it finds no major errors, it will report the number of disk pages used and the number available. If bad files are discovered, they will be listed and the system will announce "August not in operation".

8) If CHECKDISK runs successfully, the system will announce "August in operation" and ask for the date and time. Enter these in the form DD-MON-YY<SP>HH:MM; follow with <CR>. The system jobs will log in automatically.

9) When the system prompts you with "@", the prompt for EXEC, log in by typing "oper<SP>password<SP><CR>", where password stands for your password.

10) After the system prints various messages and prompts with you another "@", type "ena<CR>". The response will be a new prompt, "!".

11) Type "ref<SP>a<CR>".

6.3 Discussion

When the System XXV crashes, the first thing it does is check its recovery switches. (For information on recovery switches, see section 12, Recovery Switches.) If the recovery switches do not tell the system to come up automatically, the system simply stops and waits for someone to tell it what to do next. You now need to step in and bring the system up by providing a new copy of the monitor. In disk recovery, you do this by telling the system to get a new copy of the monitor from disk, where it is permanently stored. Disk recovery thus saves you the inconvenience of finding and loading the monitor tape and switching all the switches on the control panel. However, it will not always work. To start the copying procedure, you must use part of the old monitor called "EDDT". EDDT escapes most crashes without harm; however, crashes due to power failure always destroy EDDT and sometimes other crashes, for example, those due to power surges, will also damage it. If you suspect that the crash was due to power failure, do not try to bring up the system with disk recovery; use tape recovery instead.

You begin disk recovery by typing "dskrld<ESC>g". "Dskrld" stands for "disk reload"; it is the name of a location in the system's memory. This location is the beginning of the disk recovery procedure, a program that copies a new monitor from a file stored on the disk. When you type "dskrld<ESC>g", you tell EDDT to go to this procedure and begin running the program found there. When the procedure begins, it prints "reloading from disk". If this message never appears, it means EDDT was wiped out by the crash and you cannot reach the disk recovery procedure. In this case, begin a tape recovery.

If control is successfully transferred to the disk recovery procedure, the procedure first moves itself to a special spot in memory, beginning at location 3000, and makes room for the new monitor by clearing the rest of central memory. The system next needs to know where to should look for a new monitor. Each disk has a copy of the monitor stored in a file named <SYSTEM>MONITOR.PACK-x;1, where x stands for the disk number. The monitor file on disk pack 0, for example, is named MONITOR.PACK-0;1. To find out which disk it should check for the new monitor, the system will ask "BOOT FROM DISK PACK # [CR FOR ANY]". The standard answer here is <CR>. This tells the system to start by looking on disk pack 0 for the file; if it is not there, look on disk pack 1, and finally check disk pack 2. To tell the system to check only a particular disk pack, instead of answering the question with <CR>, give the number of the pack. If the system cannot find a good monitor file, it will print out "FAILED TO READ RESIDENT MONITOR". Since disk recovery cannot work without reading the resident monitor from disk, you will have to halt the system with Method B in section 13.5 and then use tape recovery to bring the system up.

If the system does find a usable copy of the monitor file. the disk reload procedure copies a new resident monitor from disk into the cleared memory. The system will inform you of its progress with various messages. Since you still don't really know if this procedure escaped the crash without harm, it is wise to keep an eye on these messages. If anything goes wrong before the system runs CHECKDISK and reports on the status of the file system, it means the procedure is unusable. If the system hangs as it comes up, halt it with Method B documented in section 13.5 of "Related Procedures" and bring it up with tape recovery. If the system crashes again, begin a tape recovery.

After the resident monitor is in memory, the system will go into EDDT and print "EDDT" on the operator's terminal. When you type "start<ESC>g", you transfer control to the Start procedure. This procedure starts up the rest of the recovery procedure and copies the swappable monitor from the second part of the monitor file.

As the system copies the new monitor, the old settings of the recovery switches are replaced by the default switch settings that are part of the new monitor. These default setting are: DBUGSW = 1, DCHKSW = 0, RELDSW = 1, and CDMPSW = 1. This tells the system that after crashing it should stop and wait for instructions on what to do next. Once the system is up, you may change these default switch settings with the procedure documented in section 12, Recovery Switches. That section also explains recovery switches in general.

Once the system's new monitor is in place, the remainder of disk recovery is exactly the same as tape recovery. Thus, the following explanation is identical to the last part of the Discussion in the section on tape recovery. This explanation is included here for your convenience; if you are already familiar with tape recovery, you do not need to read further.

Now that it has its new monitor, the system turns its attention to the memory and file system. It first reports on the size of the memory and tells you about the BAT blocks. "BAT" stands for "Bad Address Table". BAT blocks contain tables that are used to keep track of what parts of the disk are bad and thus should not be used. Once it is determined what parts of the disk are bad and should not be used for storage, the system runs a program called "CHECKDISK". CHECKDISK, as the name indicates, checks the disks and the integrity of the file system It makes sure that no section of the disk is allocated to more than one file and that all file addresses are valid. If CHECKDISK discovers errors in the file system, it lists the bad files, and the system stops and waits for you to correct them. In this case, the system will not be able to come up; to let you know what is happening it will announce, "August not in operation". For more information on CHECKDISK and instructions for correcting the errors it detects, see section 13.2, Correcting Problems Found by CHECKDISK.

If CHECKDISK finds no serious file problems, it reports on disk use, and then, once it is finished, the system will announce, "August in operation". At this point, the system is completely ready to come up and open itself to users. It needs only two more things from you, the date and time. When the system directs you to enter the current date and tims, type two numbers for the day, a dash, the first three letters of the month, a dash, and then two numbers for the year. Follow these with a space and then give the time, on a 24 hour basis, as two numbers for the hour, a colon, and then two numbers for the minutes; be sure to give the correct time. Follow all this with a carriage return. For example, you would enter the date March 9, 1981, and the time 5:04 pm, as "09-mar-81<SP>17:04<CR>". If you enter the wrong date and time, finish the recovery procedure and then correct your mistake as documented in section 13.6, Changing the Date and Time.

After you have entered the date and time the system is officially up. The system jobs will now log in automatically and you will be prompted with "@", the prompt for EXEC. This is an invitation to log in. Log in as an operator by typing "oper<SP>password<SP><CR>", that is: "oper" (for operator), a space, your password, a space, and then a carriage return. In the interests of secrecy, your password will not print. After you have logged in, the system will print various messages and another "@". Type "ena<CR>". This stands for "enable" and tells the system to allow you to perform operations denied the normal user. Once you have "enabled", or identified yourself to the system as a person with special powers, the system will change its prompt to "!". Now refuse automatic logout by typing "ref<SP>a<CR>". AUGUST normally logs out users who leave their terminals idle.

6.4 Errors and Recoveries

Nothing happens when you type "dskrld<ESC>g"

If nothing happens when you type "dskrld<ESC>g", this means the disk recovery procedure cannot be used. Instead, use the tape recovery procedure.

The system cannot find a monitor file

If the system cannot find the monitor file, it will tell you "FAILED TO READ RESIDENT MONITOR". If you have told the system to look on a specific disk for the monitor, halt the system (with Method B of section 13.5), try another disk recovery and tell the system to look on a different disk for the monitor. If, after checking them all (either by typing a <CR> or individually giving the number of each disk), you discover that none of the disks have a good copy of the monitor file, you cannot use disk recovery. Halt the system, if necessary, and bring it up with a tape recovery.

Errors before CHECKDISK reports on the file system

If the system hangs or crashes before CHECKDISK reports on the status of the file system and you never get the message "August in operation" or "August not in operation", recovery will not be successful. Bring up the system with the tape recovery procedure.

CHECKDISK discovers problems with the file system

If CHECKDISK finds anything wrong with the file system, the System XXV will stop and wait for you to correct the problems. It cannot come up while something is wrong with the file system: the risk of destroying files is too great. For directions on how to correct any problems CHECKDISK finds, see section 13.2, Correcting Problems Found by CHECKDISK.

7 AUTOMATIC RECOVERY

7.1 Introduction

How a System XXV responds to a crash is determined by its recovery switches. Recovery switches are explained in section 12, Recovery Switches. If they are set for automatic recovery BEFORE a crash, then, after one occurs, the system should immediately try to bring itself up with the procedure documented here. This is convenient, since you do not have to start a recovery procedure every time the system crashes. However, do not assume that systems set for automatic recovery will never need your help. If the crash was due to power failure or it damaged memory, automatic recovery will never begin; the system will simply sit there and you will have to begin a tape recovery. If the recovery procedure was somehow damaged in the crash, the system may start to bring itself up and then hang or crash again. In this case too, you must step in, halt the system, if necessary, and use tape recovery. Even if automatic recovery begins and gets as far as running CHECKDISK, success is not guaranteed. If CHECKDISK detects problems in the file system, automatic recovery can proceed no further. After correcting the file problems and halting the system (both documented in section 13, Related Procedures), you will have to use disk recovery to bring the system up.

7.2 Summary

If a System XXV set for automatic recovery comes up successfully, you do not need to do anything until you log in as an operator. If recovery never starts, if it fails before CHECKDISK reports on the file system, or if CHECKDISK discovers bad files, see "Errors and Recoveries" in this section for instructions.

1) The system will begin to bring itself up; various messages will record its progress.

2) CHECKDISK will run and check the file system. If it finds no major errors, it will report the number of disk pages used and the number available. If bad files are discovered, they will be listed and the system will announce "August not in operation".

3) If CHECKDISK runs successfully, the system will announce "August in operation". The system is now up. The system jobs will log in automatically.

4) When the system prompts you with "@", the prompt for EXEC, log in by typing "oper<SP>password<SP><CR>", where password stands for your password.

5) After system prints various messages and prompts with you another "@", type "ena<CR>". The response will be a new prompt, "!".

6) Type "ref<SP>a<CR>".

7.3 Discussion

When a System XXV crashes, the first thing it does is check its recovery switches, the switches which tell it what it should do next. Recovery switches and their various settings are explained in section 12, Recovery Switches. One setting of the recovery switches will tell the system to come up automatically. If, after a crash, the system discovers that the switches are set in this way, it will immediately start to bring itself back up without waiting for an operator.

There are, however, several problems that can stop systems from coming up automatically. First of all, the system may never find out that it was supposed to do this. In crashes due to power failure and those that damage memory, the recovery switch settings may be lost or never checked. Consequently, a system you think is set to come up automatically will not. Instead, it will wait for you to begin a recovery procedure, just as it normally does after a crash. Keep an eye on all systems set for automatic recovery; if you see a system that appears to be down and not trying to come up, you will have to use the tape recovery procedure to bring it up. Do not try to use the disk recovery procedure; it too will be lost along with the recovery switches.

If the system does remember its recovery switch settings and try to come up automatically, it usually begins by copying the current contents of central memory into two files. The first 512 pages of memory are stored in a file called <SYSTEM>CORDMP.LOW and the second 512 pages are stored in a file called <SYSTEM>CORDMP.HGH. These files are used by systems programmers to find out what was in central memory right after the crash. If you do not want the system to bother with this copying, you may set the recovery switches so that it will not be done. See section 12, Recovery Switches.

Once the contents of core have been safely stored in the CORDMP files, the system next needs a new copy of the monitor. The system copies the monitor from disk with a procedure very much like the disk recovery procedure. The procedure prints messages to help you follow its progress. It is a very good idea to read these messages and make sure recovery is progressing successfully. Even after the recovery procedure starts, things can still go wrong. If the procedure was damaged by the crash, the system may hang as it comes up or may try to come up, fail, and crash again. After crashing, the system would once more check the recovery switches, discover it should come up automatically, take another core dump, and try to come up. As it tried to come up, the system would encounter the same problem and crash again. The system could thus get caught in a loop of crashing, trying to come up, and crashing again. If you notice a system set for automatic recovery that appears to be hung or having some kind of trouble, watch it for a while. If it never gets to the point of running CHECKDISK, halt the system with Method B of section 13.5, Halting the System. Then switch to tape recovery.

If all goes well with the automatic recovery procedure, it will announce "reloading from disk", move itself to a special place in memory, starting at location 3000 and clear the rest of core. A new resident monitor is then copied from a file where it is is permanently stored on the disk. After the resident monitor is read, the system starts up and transfers control to the Start procedure. This very much like the procedure you get when you give the Start command in the disk recovery procedure. The Start procedure starts up the rest of the reocvery procedure and copies the swappable monitor from the second part of the MONITOR file.

Now that it has its new monitor, the system turns its attention to the memory and file system. It first reports on the size of the memory and tells you about the BAT blocks. "BAT" stands for "Bad Address Table". BAT blocks contain tables that are used to keep track of what parts of the disk are bad and thus should not be used. Once it is determined what parts of the disk are bad and should not be used for storage, the system runs a program called "CHECKDISK". CHECKDISK, as the name indicates, checks the disks and the integrity of the file system It makes sure that no section of the disk is allocated to more than one file and that all file addresses are valid. If CHECKDISK discovers errors in the file system, it lists the bad files, and the system stops and waits for you to correct them. Thus, if CHECKDISK discovers problems, the system cannot come all the way up automatically. Instead, the system will announce, "August not in operation" you must take over and fix the file problems CHECKDISK has found. For instructions on how to do so, see section 13.2, Correcting Errors Found by CHECKDISK.

If CHECKDISK finds nothing wrong with the file system, it reports on disk use, and then, once it is finished, the system will announce, "August in operation". At this point, the system is up. Notice that you are not required to enter the date and time as you must do to end the disk and tape recovery procedures. During automatic recovery, unlike the other recovery procedures, the system's internal clock continues to run. To learn the correct time, the system simply uses it instead of asking you. In addition to using the system's clock to find the time, the end of the automatic recovery procedure differs in another way from disk and tape recovery. During automatic recovery, the system saves the original recovery switches settings. When recovery is over, these settings are restored and replace the default switches settings that are read in as part of the new monitor. This means that after an automatic recovery the recovery switches continue to be set for automatic: next time the system crashes it will again try to bring itself up automatically.

Once the system has found out the time and come all the way up, the system jobs can log in automatically and you will be prompted with "@", the herald for EXEC. This is an invitation to log in. Log in as an operator by typing "oper<SP>password<SP><CR>", that is: "oper" (for operator), a space, your password, a space, and then a carriage return. In the interests of secrecy, your password will not print. After you have logged in, the system will print various messages and another "@". Type "ena<CR>". This stands for "enable" and tells the system to allow you to perform operations denied the normal user. Once you have "enabled", or identified yourself to the system as a person with special powers, the system will change its prompt to "!". Now refuse automatic logout by typing "ref<SP>a<CR>". AUGUST normally logs out users who leave their terminals idle.

7.4 Errors and Recoveries

The System never begins to bring itself up

If a system that is supposed to be set for automatic recovery never announces "reloading from disk" to show it has begun, bring the system up with tape recovery.

Errors before CHECKDISK reports on the file system

If the system hangs or crashes before CHECKDISK reports on the status of the file system and you never get the message "August in operation" or "August not in operation", recovery will not be successful. Bring up the system with the tape recovery procedure.

CHECKDISK discovers problems with the file system

If CHECKDISK finds anything wrong with the file system, the System XXV will come up automatically; the risk of destroying files is too great. Instead, it will stop and wait for you to correct the problems. For directions on how to correct any problems CHECKDISK finds, see section 13.2, Correcting Problems Found by CHECKDISK.

8 TAPE RECOVERY

8.1 Introduction

Tape recovery is basically a backup recovery procedure. It should be used after automatic or disk recovery has failed or after a crash due to power failure. Tape recovery is normally a straightforward and not particularly difficult procedure, but it can grow rather complicated, particularly if the crash has somehow damaged the file system or wreaked other havoc. If you encounter problems while bringing the system up, check section 8.4, Errors and Recoveries -- you may find the solution to your problem there. If your problem is not covered in Errors and Recoveries, try the whole procedure over, starting from step 3. If this does not work, something may be seriously wrong. Notify your manager; he or she may want to try Standalone Recovery or, as a last resort, Disk Rebuild.

8.2 Summary

1) Check the BUGHLT number on the operator's terminal and look it up in the list of BUGHLTs; in addition, note which error lights are lit. Record all this information.

2) If the power has gone off, you must reload the microcode as documented in steps 3 through 12. If the power has not gone off, skip to step 13.

3) Mount the microcode tape on the tape drive.

4) Put all switches on the control panel off (down).

5) Put address switch 32 on (up).

6) Put MICRO PROCESSOR STOP on.

7) Put MICRO PROCESSOR MIPC on.

8) Put MICRO PROCESSOR CLR momentarily on.

9) Put MICRO PROCESSOR CONT momentarily on.

10) Put MICRO PROCESSOR MIPC off.

11) Put MICRO PROCESSOR STOP off.

12) Put MICRO PROCESSOR CONT momentarily on. The tape should spin and then stop. Remove the microcode tape from the tape drive.

13) You are now ready to read in the new monitor. Mount the monitor tape on the tape drive. [Start here if you do not want to load the microcode.]

14) Put all switches on the control panel off.

15) Put address switches 24 and 26 on.

16) Put MICRO PROCESSOR STOP on.

17) Put MICRO PROCESSOR MIPC on.

18) Put MICRO PROCESSOR CLR momentarily on.

19) Put MICRO PROCESSOR CONT momentarily on.

20) Put MICRO PROCESSOR MIPC off.

21) Put MICRO PROCESSOR STOP off.

22) Momentarily put MICRO PROCESSOR CONT on. The tape should spin and then stop.

23) Put address switches 24 and 26 off.

24) Put address switches 29 and 30 on.

25) Momentarily put CONSOLE START on twice.

26) When the operator's terminal says "EDDT", type "start<ESC>g". The monitor tape should spin.

27) Remove the monitor tape from the tape drive.

28) After the system reports the size of the memory, put MI PAR ERR STOP and MEM PAR ERR STOP on. The system will print several messages about BAT blocks.

29) CHECKDISK will run and check the file system. If it finds no major errors, it will report the number of disk pages used and the number available. If bad files are discovered, they will be listed and the system will announce "August not in operation".

30) If CHECKDISK runs successfully, the system will announce "August in operation" and ask for the date and time. Enter these in the form DD-MON-YY<SP>HH:MM; follow with <CR>. The system jobs will log in automatically.

31) When the system prompts you with "@", the prompt for EXEC, log in by typing "oper<SP>password<SP><CR>", where password stands for your password.

32) After system prints various messages and prompts with you another "@", type "ena<CR>". The response will be a new prompt, "!".

33) Type "ref<SP>a<CR>".

8.3 Discussion

Although you can bring System XXV up after most crashes simply by providing it with a new monitor, this is not always true. Some crashes, especially those due to power failure, wipe out not only the monitor, but also the microcode. Since the microcode is the information the microprocessor uses to translate the monitor instruction into something it can understand, when this happens you must begin recovery by reading in a tape containing the microcode. Only after the mocrocode is available can the system correctly read the monitor tape.

Since a system that does not have access to microcode or any part of the monitor cannot understand any instructions given on the operator's terminal, to reload the microcode you must give the system instructions from the control panel. Mount the microcode tape on the tape drive and put address switch 32 on by pushing the switch up. Also, find the row of switches labeled "MICRO" and put the MICRO PROCESSOR STOP and MICRO PROCESSOR MIPC on. You then briefly put on MICRO PROCESSOR CLR followed by then MICRO PROCESSOR CONT. This process tells the microprocessor that the address specified through the address switches is where it should look for instructions on what to do next. The address you specify by putting on address switch 32 is address 10 octal. This is the beginning of a tape-reading routine which is permanently stored in the memory of the microprocessor. You now want to tell the microprocessor to execute this routine and read the tape containing the microcode. You do this by putting off MICRO PROCESSOR MIPC and MICRO PROCESSOR STOP and putting on MICRO PROCESSOR CONT.

Once the system has read the tape containing the microcode, it has all the information necessary to read the first part of the monitor tape, which contains the resident monitor. The procedure for reading this tape is identical to that for reading the microcode tape, EXCEPT that you specify a different address with the address switches. First put off all the switches on the control panel, then put on address switches 24 and 26, put on MICRO PROCESSOR STOP and MICRO PROCESSOR MIPC, and finally, again momentarily put on MICRO PROCESSOR CLR and MICRO PROCESSOR CONT. This tells the system it should begin executing the instructions at address 5000 octal, the address specified with address switches 24 and 26. Address 5000 is the beginning of instructions for reading the monitor. To execute these instructions, put MICRO PROCESSOR MIPC and MICRO PROCESSOR STOP off and again momentarily put on MICRO PROCESSOR CONT. The microprocessor will read into memory the first part of the monitor tape: this contains the resident monitor.

Once the resident monitor is in memory, you want to start it running. When you put on address switches 29 and 30 and then hit CONSOLE START twice, you tell the system to go to location 140 and start running the procedure it finds there. The procedure beginning at this location brings the system alive and starts up the resident monitor. Because EDDT is part of the resident monitor, the system can now go into EDDT and will print "EDDT" on the operator's terminal to notify you. Since EDDT can understand typed commands, you can start up the rest of the recovery procedure by typing "start<esc>g" on the operator's terminal. The system will begin running and read in the swappable monitor, the second part of the monitor tape.

When the system copies the new monitor, the old settings of the recovery switches are changed to the default switch settings that are part of the new monitor. These default settings are: DBUGSW = 1, DCHKSW = 0, RELDSW = 1, and CDMPSW = 1. This tells the system that after crashing it should stop and wait for instructions on what to do next. Once the system is up, you may change these default switch settings with the procedure documented in section 12, Recovery Switches. That section also explains recovery switches in general.

Now that it has its new monitor, the system turns its attention to the memory and file system. It first checks how much memory is physically available and reports on the size of the memory. Putting on MI PAR ERR STOP and MEM PAR ERR STOP tells the system to stop if a parity error is encountered in central memory or in the microcode. The system will next tell you about the BAT blocks. "BAT" stands for "Bad Address Table". BAT blocks contain tables that are used to keep track of what parts of the disk are bad and thus should not be used. Once it is determined what parts of the disk are bad and should not be used for storage, the system runs a program called "CHECKDISK". CHECKDISK, as the name indicates, checks the disks and the integrity of the file system It makes sure that no section of the disk is allocated to more than one file and that all file addresses are valid. If CHECKDISK discovers errors in the file system, it lists the bad files, and the system stops and waits for you to correct them. In this case, the system will not be able to come up. Instead, after CHECKDISK runs, the system will announce, "August not in operation". For more information on CHECKDISK and instructions for correcting the errors it detects, see section 13.2, Correcting Problems Found by CHECKDISK.

If CHECKDISK finds no serious file problems, it reports on disk use, and then, once it is finished, the system will announce, "August in operation". At this point, the system is completely ready to come up and open itself to users. It needs only two more things from you, the date and time. When the system directs you to enter the current date and tims, type two numbers for the day, a dash, the first three letters of the month, a dash, and then two numbers for the year. Follow these with a space and then give the time, on a 24 hour basis, as two numbers for the hour, a colon, and then two numbers for the minutes: be sure to give the correct time. Follow all this with a carriage return. For example, you would enter the date March 9, 1981, and the time 5:04 pm. as "09-mar-81<SP>17:04<CR>". If you enter the wrong date and time, finish the recovery procedure and then correct your mistake as documented in section 13.6, Changing the Date and Time.

After you have entered the date and time the system is officially up. The system jobs will now log in automatically and you will be prompted with "@", the prompt for EXEC. This is an invitation to log in. Log in as an operator by typing "oper<SP>password<SP><CR>", that is: "oper" (for operator), a space, your password, a space, and then a carriage return. In the interests of secrecy, your password will not print. After you have logged in, the system will print various messages and another "@". Type "ena<CR>". This stands for "enable" and tells the system to allow you to perform operations denied the normal user. Once you have "enabled", or identified yourself to the system as a person with special powers, the system will change its prompt to "!". Now refuse automatic logout by typing "ref<SP>a<CR>". AUGUST normally logs out users who leave their terminals idle.

8.4 Errors and Recoveries

Using an old monitor tape

You may sometimes have to perform a tape recovery with an old monitor tape, for example, when you do not have a copy of the current monitor or the current tape is bad. When this happens, you can use the resident monitor from an old tape to start the system runnin. Once the system is up, you can switch to disk recovery to replace the old resident monitor with a good copy of the monitor taken from disk. The procedure is as follows:

1) Follow the tape recovery procedure from step 13 through 25. If there has been a power, problem, do step 3 through 25.

2) When the system types "EDDT" on the operator's terminal, type "dskrld<ESC>g" to start a disk recovery.

3) Follow the disk recovery procedure beginning from step 3.

Problems reading the microcode tape

If you cannot read the microcode tape, there is a hardware problem. Call Tymshare Maintenance.

Problems reading the monitor tape

If you cannot read the monitor tape, the microcode may have been destroyed. Start the recovery procedure over and begin at step 3 by loading the microcode. If you do not succeed, call Tymshare Maintenance.

The interrupt message

If you get a message, "Interrupt at nnn.", where nnn is some number, try reading both tapes again. If you are unsuccessful, call Tymshare Maintenance.

CHECKDISK is never run

If the system hangs or crashes before CHECKDISK reports on the status of the file system and you never get the message "August in operation" or "August not in operation", recovery will not be successful. Halt the system, if necessary, and try the recovery procedure over from step 3. If the complete procedure does not work on the third try, call Tymshare Maintenance.

CHECKDISK discovers problems with the file system

If CHECKDISK finds anything wrong with the file system, the System XXV will stop and wait for you to correct the problems. It cannot come up while something is wrong with the file system: the risk of destroying files is too great. For directions on how to correct any problems CHECKDISK finds, see section 13.2, Correcting Problems Found by CHECKDISK.

9 STANDALONE RECOVERY

9.1 Introduction

Bringing the system up with the standalone recovery procedure is useful when some error in the disk, CHECKDISK, or the system jobs is causing the system to crash before it can come all the way up. During a standalone recovery, the system does not run CHECKDISK and the other checking programs that are part of the three "normal" recovery procedures. Instead, the system comes up without checking itself and, after it is up, is shut to normal users; only the person at the operator's terminal is allowed in. Standalone recovery is risky. Do not bring the system up with this procedure unless specifically instructed to do so.

9.2 Summary

1) Check the BUGHLT number on the operator's terminal and look it up in the list of BUGHLTs; in addition, note which error lights are lit. Record all this information.

2) Follow the procedure for tape recovery (section 8) from step 13 through step 25. If you suspect there has been a power failure, do step 3 through 25 of the tape recovery procedure.

3) When the operator's terminal says "EDDT", type "dbugsw/". The system will print either 0 (zero) or 1.

4) Type "2<CR>".

5) Type "start<ESC>g". The monitor tape should spin.

6) The system will request the date and time. Enter these as DD-MON-YY<SP>HH:MM and follow with <CR>.

7) You will automatically be logged in as "system", but not enabled.

9.3 Discussion

To use standalone recovery procedure, you begin by following the tape recovery procedure. But after the system reads the first part of the monitor tape and tells you it is in EDDT, you do NOT type "start<esc>g", to start the system running and read in the rest of the tape. Instead, you work in EDDT an interactive language for debugging. EDDT is part of the resident monitor and is used to patch and otherwise manipulate it. Because it can change the monitor, EDDT is a very powerful tool; use it with care.

Once the system tells you that you are in EDDT, do not wait to be prompted. EDDT has no prompt; as you use it, it simply waits for you to type something and then reacts. When you see that are in EDDT, immediately type "dbugsw/". This command has two parts. The first part, "dbugsw", is the name of an address. The "/" means "print". Thus, "dbugsw/" instructs the system to print what it finds at the location DBUGSW. This location contains one of the system's recovery switches, the debugging switch. The number at this address tells the system what it should do when it encounters a fatal error. A zero (0) at DBUGSW means the system should respond to errors by crashing. A 1 means the system should take breakpoints; that is, when a fatal error occurs the system should not crash but should stop where it is, preserve the context of the error, and print out a BUGHLT address. This address is what you record after a crash when you are instructed to record the BUGHLT number. Knowing the address of the error that caused the system to crash helps systems programmers find out what happened. Recovery switches are further explained in section 12, Recovery Switches.

After you print the current contents of DBUGSW, type "2<CR>. This tells the system to enter 2 at this location. When DBUGSW is 2, it instructs the system to skip running CHECKDISK and the system jobs, and to come up "standalone". When a system comes up standalone, it accepts input only from the operator's terminal; it does not allow any ordinary users to log in.

Once you have made sure the system will come up isolated from the outside world, you start it by typing "start<ESC>g". The system will read the second part of the monitor tape, the part containing swappable monitor, and ask you for the date and time. After you have entered these (as DD-MON-YY<SP>HH:MM<CR>), the system will come up and automatically log you in as "system". This automatic login keeps the system from going through the complicated login procedure. When you are logged in as "system", you have the same powers as if you had logged in as "operator"; remember to enable if you want to do anything requiring special powers.

10 DISK REBUILD STRATEGY OR TOTAL CATASTROPHE

10.1 Introduction

Occasionally, a particularly deadly system crash destroys the file system. If you suspect this has happened, immediately notify your manager and, if possible, an OAD operating systems programmer; do not attempt to do more.

When the file system is destroyed, you must use the various dumps made each week to rebuild the disk and restore the files as completely as possible to their pre-crash state. This must be finished before the system needs anything from the disk. Rebuilding the disk is a fairly simple procedure, but the loss of users' files and the possibility that they may be damaged or incompletely restored is so serious that you should NEVER undertake a disk rebuild without specific instructions and assistance of a manager or an operating systems programmer.

10.2 Summary

WARNING: Never attempt this without specific instructions from a manager or an operating systems programmer.

1) Check with Tymshare Maintenance to make sure the hardware is good.

2) Follow the tape recovery procedure from steps 13 through 25. If there has been a power failure, do steps 1 through 25.

3) When the operator's terminal says "EDDT", type "dbugsw/". The system will print either zero (0) or one (1).

4) Type "2<CR>".

5) Type "syslod<ESC>g".

6) The system will ask, "Do you really want to clobber the disk by reinitializing?".

7) Type "y<CR>". This stands for "yes". Do not type more than y.

8) The system will say, "OK, You asked for it..."

9) The system will reinitialize all the files and then report, "No EXEC".

10) Load the DLUSER tape on the tape drive.

11) Type "l", for "load". When the system asks, "Load from magtape MTAN:", type "mta0:<CR>" ("0" here is zero) and confirm with another <CR>.

12) When the system asks, "File Number?", type "0<CR>".

13) The system will now read the DLUSER file, the first part of DLUSER tape.

14) When it has finished, the system will prompt you with a period (.), the prompt for MINI-EXEC. At the period, type "s.".

15) The system will print, "Interrupt at nnn", where nnn is some number, followed by a period.

16) To read the second file in the tape, the DUMPER file, type "l", for "load". When the system asks, "Load from magtape MTAN:", type "mta0:<CR>", and confirm with another <CR>.

17) When the system asks, "File Number?", type "1<CR>".

18) The system will now read the DUMPER file.

19) When the system prompts you with a period, type "s.".

20) The program DUMPER is now loaded and ready to start restoring the files. Mount the first Full Dump Tape. Make sure you load the Full Dump Tapes in numerical order.

21) DUMPER will now ask a series of questions, preceding each of them with instructions.

22) To answer the first question, "DUMP, LOAD, CHECK, OR SINGLE?", type "l", for "load".

23) For the second question, "DO YOU WISH TO SUPERSEDE OLDER VERSIONS ALWAYS?", type "n", for "no".

24) When DUMPER asks, "SPECIFIC USERS?", type "n".

25) When it asks, "INTO SAME DIRECTORIES?", type "y".

26) Finally, when requested, "TYPE MAG TAPE UNIT NUMBER", type "0" (zero).

27) DUMPER will now read the tape; when it is finished, it will print, "MOUNT NEXT TAPE, IF ANY. TYPE C, WHEN READY, N, IF NO MORE". Mount the next tape and type "c".

28) DUMPER will again ask for the mag tape unit number; type "0".

29) Continue mounting and loading Full Dump Tapes, typing "c" to continue and then giving 0 (zero) for mag tape unit number, until all the tapes have been read.

30) When you have finished loading the Full Dump Tapes, begin loading the Incremental Dump Tapes. Be sure that you load the Incremental Dump Tapes in chronological order, beginning with the one made right after the Full Dump and ending with the most recent.

31) After the system has read the last Incremental Dump Tape, when DUMPER says, "MOUNT NEXT TAPE, IF ANY. TYPE C, WHEN READY, N, IF NO MORE", type "n".

32) DUMPER will stop and the system will print an interrupt message followed by a period, the prompt for MINI-EXEC.

33) The files have now been restored as completely as possible. Halt the system, and begin a disk recovery. (To halt the system use Method A of section 13.5, Halting the System.)

10.3 Discussion

A disk rebuild is necessary when a crash destroys the files. Since a system that has lost its files cannot be expected to run CHECKDISK or the system jobs successfully -- these are stored on the disk and everything on the disk has been lost -- the system must be brought up in such a way that it does not need anything from its files; in fact, it does not even realize they are lost. This means the system must be brought up standalone, since in a standalone recovery the system skips running the system jobs, does not check the file system with CHECKDISK , and does not open itself for normal use.

But bringing up the system for a disk rebuild is not a completely "normal" example of bringing the system up standalone. After you have set DBUGSW to 2 (to make the system come up without checking itself and closed to users), you do not then type "start<ESC>g" to start the system running. Instead you type "syslod<ESC>g". This stands for "system load". It tells the system to begin running a program that wipes out all existing files and then allows you rebuild the entire file system with files copied from tape.

When you give the Syslod command, the system will ask, "Do you really want to clobber the file system?". When you respond "y", for "yes", it will print, "OK, you asked for it..." and reinitialize all the files. When the files have been reinitialized, the system will state: "No EXEC". EXEC disappears because it was stored on the disk. After informing you of EXEC's disappearance, the system will prompt you with a period (.), the prompt for MINI-EXEC. MINI-EXEC is a group of basic commands that are loaded with the monitor. MINI-EXEC has two important features: First, MINI-EXEC recognizes commands by their first letter, so you type "l" for "load", "s" for "start", and so on; and second, in MINI-EXEC you must end all simple commands that do not ask you for further information with a period for confirmation.

NOTE: Because MINI-EXEC recognizes commands by their first letter, if you make a mistake in giving a simple command, type a few more random characters before you confirm with a period. The additional letters you type will make the command unrecognizable. When you are again prompted with a period, repeat the command you want. If you are giving a command that asks you for further information before it is executed, type some random characters in answer to the additional question. This will cause the command to be aborted and the period will reappear.

Once you are in MINI-EXEC, you can begin the procedure for rebuilding the disk from backup tapes. Mount the DLUSER tape on tape drive zero, and type "l", for "load". When the system asks you, "Load from magtape MTAN:", identify your drive as "mta0:" and confirm by typing "<CR><CR>". The system will now ask which file on the tape it should read by printing: "File Number?". The number of the DLUSER file, which should be printed on the tape casing, is zero (0). Enter this and follow it with a carriage return.

DLUSER stands for "dump and load users". The DLUSER file contains data about all the directories on the system, both the user and system directories, and a program that can use this information to rebuild them. When you type "s.", you instruct the system to run this program.

When the system has rebuilt the directories, it will print an interrupt message. This means it is ready to read another file. You now want to load the file containing the DUMPER program. To do this again, type "l", and then again, when asked, "Load from magtape MTAN:", identify your drive as "mta0:" and confirm by typing "<CR><CR>". Next, you will be asked for the file number. The file number for the DUMPER file is one (1); this also should be printed on the tape casing. Enter 1 and follow it with a confirming <CR>. The DUMPER file contains a program able to read files from tape and restore them to the correct directories. Once you have loaded DUMPER and typed "s." to start it, the DUMPER program will start running. You can now use this program to rebuild the disk by mounting and reading back into the system the dump tapes that contain the back-up versions of all the files on the disk.

To ascertain what it should do, DUMPER will ask you a series of questions. Each question is preceded by an explanation of how you should answer it. These explanations are designed for people using DUMPER for routine maintenance of the file system. Do not be alarmed if the answers you are instructed to give here do not agree with what the system tells you to do. Disk rebuild is not a normal situation.

The first thing DUMPER will want to know is what you plan to do. To find out, DUMPER will ask: "DUMP, LOAD, CHECK, OR SINGLE?". Since you want to load files from tape back into the disk type "l", for "load". Then mount the first Full Dump Tape on the tape drive. Full Dump Tapes are tapes made at regular intervals, usually weekly, that contain a record of the entire contents of the disk. The first Full Dump Tape contains all the information the system needs to run. As you mount this and the following tapes on the tape drive, make sure they do not have write rings.

DUMPER now will try to find out what to do with the information on the tape. It will first ask "DO YOU WANT TO SUPERSEDE OLDER VERSIONS ALWAYS?". Answer with a "n", for "no". This makes sure that DUMPER will put the files from tape and the files already on the disk in the correct order and pay attention to version numbers. DUMPER will then ask "SPECIFIC USERS?". DUMPER asks this because it normally restores the files of single users whose directories are somehow lost or mutilated. Since you want to restore the all the files of every user, type "n", for "no"; and, when DUMPER wants to know: "INTO SAME DIRECTORIES?", type "y". DUMPER's final request will be: "TYPE MAG TAPE UNIT NUMBER". After you type "0", DUMPER will copy the files from the currently mounted tape into their directories. When it has finished, it will print: "MOUNT NEXT TAPE, IF ANY; TYPE C WHEN READY, N, IF NO MORE". Load the next full dump tape, making sure it does not have a write ring, and type "c", for "continue". When the mag tape unit number is requested, answer with "0". This new tape will then be read, and DUMPER will again ask if you want to go on. Continue loading the Full Dump Tapes until all have been read.

When you have finished loading the Full Dump Tapes, it is time to load the Incremental Dump Tapes. Incremental Dump Tapes are tapes made every night that contain only files altered during the preceding day. As you enter the Incremental Dump Tapes, you progressively update the files entered from the Full Dump Tapes. Enter the Incremental Dump Tapes in chronological order, beginning with the tape made right after the full dump, and ending with the tape made most recently. Use the same procedure you used to load the Full Dump Tapes: mount the tape, type "c", and enter the unit number. When you have loaded the final, i.e. the most recent, Incremental Dump Tape, you will have restored the files as well as they can be restored. At this point, answer "n", for no, to DUMPER's question about any further tapes. DUMPER then will halt, and the system will print an interrupt message followed by a period, the prompt for

MINI-EXEC. Now halt the system (with Method A of section 13.5, Halting the System) and bring it up with a disk recovery.

10.4 Errors and Recoveries

Inability to read the DLUSER tape

If you are unable to read the DLUSER tape, halt the system and start the entire procedure over again.

Inability to read the first Full Dump Tape (the tape after the DLUSER tape)

Since the DLUSER tape contains a copy of DUMPER, once you have read this tape, DUMPER is stored on the disk. If you then cannot read the second tape, that is, the first Full Dump Tape, type <CTRL-P>. (You may have to do this several times.) You will get a period, the prompt for MINI-EXEC. After you have the period, halt the system and bring it up as documented in the section "Standalone Recovery". When the system is up, you can run DUMPER from disk by typing "dumper<CR>" at the EXEC "@". Once DUMPER is running, start from step 20 in the procedure documented above. If you still cannot read the first Full Dump Tape, halt the system and start the whole process again from step 1.

11 RECOVERY FROM MEMORY PARITY ERRORS

11.1 Introduction

System XXVs run on odd parity. This means that for every word of memory, the sum of the bits turned on plus the parity bit must be odd. The system checks the parity whenever it uses stored information. If it finds a word with even parity, a parity error occurs. If the system discovers a parity error, it first tries to correct the error itself. If the error cannot be corrected, then the system automatically scans core, prints an error message listing the locations and contents of the offending addresses, and stops with a BUGHLT. Tymshare Maintenance must be called for all System XXV parity errors, as they indicate that the memory hardware may be bad.

11.2 Summary

Call Tymshare Maintenance for all parity errors.

12 RECOVERY SWITCHES

12.1 Introduction

The System XXV has four "recovery switches" that tell it how to respond to system errors and what to do when it crashes. These switches are actually four locations in the system's central memory, each controlling a particular aspect of the system's response. Every location or switch can have at least two different values. Changing the values changes what the system will do in the particular situation that the switch controls. For example, the switch controlling what the system does when it encounters a BUGCHK, a less serious error than a BUGHLT, can be set to 0 (zero) or 1 (one). When the switch has the value 0, the system ignores BUGCHKs, when it is set to 1, the system crashes when it encounters a BUGCHK. Thus, to make the system run as you wish, you simply set each switch to the appropriate value. The rest of this section will help you discover what this value may be and teach you how to set it. The first part, "Switches and Their Settings", discusses each of the four switches and what they control, and describes the effects of their different settings. The second part, "How to Change Switch Settings", explains how to set a switch to have the value you want.

12.2 Switches and Their Settings

The System XXV's four recovery switches are:

DBUGSW, which controls reponse to a BUGHLT

DCHKSW, which controls the response to BUGCHK

CDMPSW, which tells the system whether or not to take a core dump

RELDSW, which tells the system whether or not to actually begin automatic recovery

The system checks DCHKSW when it encounters a BUGCHK, a relatively minor type of error. The system then immediately does as this switch instructs it; no other switches are looked at. When the system encounters a BUGHLT, a fatal error, it checks DBUGSW. DBUGSW may then tell it to check the two remaining switches, CDMPSW and RELDSW. If DBUGSW does not instruct the system to look at CDMPSW and RELDSW, they are never checked.

The table below outlines the values each recovery switch can have and the effect of setting the switch to this value. The first column gives the name of the switch, the second column lists the possible values for this switch, and the third column describes how the system will act when the switch has this value. The "normal" value for each switch is marked with a stars (*). When all switches have their normal values, the system prints messages at BUGCHKs, and crashes at BUGHLTs. After the system crashes, it will wait for an operator to bring it up. If you want the system to recover automatically, rather than wait for operator's instructions, simply change the setting of DBUGSW from 1 to 0. See the next section for instructions.

Switch	Value (*=normal)	Effects
DBUGSW	0	Stop at BUGHLTs, check CDMPSW and RELDSW and do what they say. (For Automatic Recovery)
	1*	Stop at BUGHLTs, don't check CDMPSW and RELDSW, go into EDDT, and wait for an operator to begin recovery. (For Disk or Tape Recovery)
	2	Stop at BUGHLTs, don't check CDMPSW and RELDSW, go into EDDT. Don't run system-checking programs and come up shut. (For Standalone Recovery)
DCHKSW	0*	Don't stop at BUGCHKs, print error message and continue.
	1	Stop at BUGCHKs, print error message, go into EDDT, and wait for an operator to begin recovery.
CDMPSW	0	Don't take core dump before beginning recovery.
	1*	Take core dump before beginning recovery.
RELDSW	0	Don't begin automatic recovery after crashing.
	1*	Begin automatic recovery after crashing.

12.3 Switch Descriptions

DBUGSW: DBUGSW, located at memory location 76, is the switch the system checks when it encounters a BUGHLT while running. A BUGHLT is a serious system error. The system must crash when a BUGHLT occurs; this switch determines what the system does after the crash. DBUGSW can be set to 0, 1, or 2. A 0 at DBUGSW is the setting for automatic recovery. It instructs the system to check the switches CDMPSW and RELDSW and do as they say. CDMPSW will tell it whether a core dump should be taken; RELDSW will tell the system whether to actually start the recovery. (See below and the section on automatic recovery for details.) A 1 at DBUGSW is the standard setting. With this setting, upon encountering a BUGHLT, the system stops where it is, prints out a BUGHLT message, goes into EDDT, and waits for instructions from the operator's terminal. You can then begin whatever recovery procedure is appropriate. A 2 at DBUGSW has the same effect as a 1, and, in addition, after a recovery procedure is started, causes the system to come up standalone. The system will come up without running CHECKDISK and the system jobs and, after it is up, only the person at the operator's terminal is allowed access. (To do a standalone recovery, you put a 2 in DBUGSW before the system comes all the way up.)

DCHKSW: DCHKSW, located at memory address 77, is the only switch checked when the system encounters a BUGCHK: no other switches are consulted. DCHKSW can be set to 0 or 1. When DCHKSW is 0, the system will print out a BUGCHK message and then continue running. When DCHKSW is 1, the system will print out a BUGCHK message and then stop, go into EDDT, and wait for further instructions about how to come up. Since BUGCHKs are not serious errors, the usual setting for DCHKSW is 0.

CDMPSW: CDMPSW is located at memory address 100. It is checked only when a BUGHLT occurs and the system finds that DBUGSW is set to zero, the setting for automatic recovery. CDMPSW tells the system whether or not to make a copy of the contents of central memory before beginning to come up. The process of copying the contents of memory is called "taking a core dump". If CDMPSW switch is set to 0, then the system will not take a core dump. It will simply check RELDSW to see if it really should come up automatically. If CDMPSW is set to 1, before checking RELDSW, the system will copy system first 512 pages of memory into a file called <SYSTEM>CORDMP.LOW and the second 512 pages into a file called <SYSTEM>CORDMP.HGH. Since systems programmers may need to look at the contents of the memory to investigate the crash, CDMPSW is generally set to 1.

RELDSW: RELDSW is located at memory address 101. It, like CDMPSW, is checked only after a BUGHLT occurs and DBUGSW set to zero, the setting for automatic recovery. RELDSW tells the system whether or not it should actually begin this automatic recovery. A 0 in RELDSW tells the system not to recover automatically: the system will then wait for instructions from the operator's terminal just as if DBUGSW were set to 1. A 1 tells the system "yes, do begin to come up automatically". Because when DBUGSW is 0 you usually do want the system to recover automatically, the normal setting of RELDSW is 1.

12.4 How to Change Switch Settings

Introduction

This section tells you how to change the values of the System XXV's four recovery switches. To do this the system must be running correctly and you must be able to enable. To learn the names of the recovery switches, their values, and what they mean, see the previous section.

Summary

1) Check your prompt. If it is an exclamation mark (!), you are enabled. If it is not, type "ena<CR>" at the EXEC "@".

2) At the "!" prompt, type "mddt<CR>".

3) Type "switchname/", where switchname stands for the name of the switch you want to change.

4) The system will give you the current value of the switch.

5) Type "switchvalue<CR>", where switchvalue stands for the new value the switch should have: 0, 1, or 2.

6) To check the new value, type "switchname/" again. The system should show you the new value.

7) Type "<CTRL-C>"; you should return to EXEC and get the "!" prompt.

Discussion

To change the values of the recovery switches, you need to work in MDDT an interactive language for debugging. It is part of the resident monitor and is used to change and manipulate it. To enter MDDT, you first need to make sure you are enabled. Check your prompt: if it is an exclamation mark (!), you are enabled. If it is anything else, type "ena<CR>" at EXEC "@" prompt. After you are sure you are enabled, enter MDDT by typing "mddt<CR>". The system will print "mddt", to show you have entered, and then do nothing more. Like EDDT, MDDT has no herald; as you use it, it simply waits for you to tell it something and then reacts.

When you are in MDDT, to go to the switch you want to change and look at its current value, type the switch name followed by a slash (/), for example, "dchksw/". This command has two parts. The first part, "dchksw", is the name of the address that contains the recovery switch value. The "/" means "print". Thus, "dchksw/" instructs the system to show you the contents of the location "dchksw".

Once MDDT has shown you the value of the switch, it waits at this location to see if you want to do anything else. If you decide you do not want to change this switch, simply type a carriage return. To enter a different value in this address, type the value you want followed by a carriage return. The number you type will immediately become the new value of the switch. To make sure that you entered the value you wanted, again type the switch name followed by a slash. If the value is correct, simply type a carriage return. This means you are finished working with this address. If it is not correct, type the correct value and then a carriage return.

After you have changed as many of the four recovery switches as you wish, you are ready to leave MDDT. To do this, type <CTRL-C> and you will returned to EXEC.

13 RELATED PROCEDURES

13.1 Introduction

This section describes the following procedures.

Correcting Problems Found by CHECKDISK, section 13.2

Running CHECKDISK Yourself from EXEC, section 13.3

Deleting and Expunging Files, section 13.4

Halting the System, section 13.5

Changing the Date and Time, section 13.6

Connecting to and Disconnecting from the Micronode TYMBASE, section 13.7

When a recovery process requires that one of these procedures be used, you will be referred here. If you find that you never have to use any of them, do not be alarmed. This is a sign of success. These procedures are used only when something goes wrong -- when, for example, CHECKDISK finds file problems that must be corrected, you need to halt the system, or the system some how comes up with the wrong date and time.

13.2 Correcting Problems Found by CHECKDISK

Introduction

CHECKDISK is a program the system uses to check the file system before it comes all the way up and opens itself to users. If it finds any problems, the system states, "August not in operation" and stops to waits for them to be corrected with the procedure documented below. Once this is done, halt the system as documented later in "Related Procedures", and then bring it up again with disk recovery. This section deals only with recovery from file errors detected by CHECKDISK. It assumes that CHECKDISK has been run automatically. To learn how to run CHECKDISK manually, see 13.3, Running CHECKDISK Yourself from EXEC.

Summary

CHECKDISK checks the files for Illegal Disk Addresses (IDAs), Multiple Disk Addresses (MDAs) and Bit Table Errors (BTEs). If it finds any of these, it lists the files involved and their errors. To correct the problems found by CHECKDISK do the following:

1) If only one file has errors, delete and expunge that file. Be sure to type the entire file name, including all extensions; do not use <ESC> to fill out names. The

process of deleting and expunging files is described in section 13.4, Deleting and Expunging Files.

2) If more than one file is involved, delete and expunge all files with IDAs. Do not delete the files with MDAs at this point.

3) Halt the system with Method A of section 13.5, Halting the System. Then bring it up again with disk recovery. If CHECKDISK again finds files with IDAs, repeat this procedure.

4) Once no files have IDAs, if one or more files have MDAs, delete and expunge the file with the largest number of MDAs, halt the system, and bring it up with disk recovery. Do this three times. If you then still have more than 20 files with MDAs, call an operating systems programmer.

NOTE: Keep a list of the files you delete and expunge, and restore them after the system comes up. Always send messages to all users whose files have been deleted and restored.

Discussion

CHECKDISK can detect three types of errors: Bit Table Errors (BTEs), Illegal Disk Addresses (IDAs), and Multiple Disk Addresses (MDAs). CHECKDISK can correct BTEs without assistance. It cannot, however, correct IDAs or MDAs. These two errors are what are known as Page Table Errors. They occur when the system's file map, stored in what is called a "page table", is incorrect. AUGUST memory is divided into units called pages, each consisting of 512 words. File storage is allotted by pages, and one page is the smallest unit of storage that can be transferred from disk to core. A page table is like a table of contents for the disk storage. For each file, it records the addresses of all the pages allocated to that file.) An IDA means there is a disk address that is garbage. An MDA means the system has assigned the same part of the disk to two or more files. If these errors are allowed to go uncorrected, they can destroy the file system.

The remedy for problems detected by CHECKDISK is to delete the files that really do have bad storage addresses. If there is only one file with bad addresses, there is not a serious problem; simply delete that file. If more than one file is afflicted, begin by deleting all files with IDAs. IDAs are a frequent cause of MDAs. Often, when the system follows an IDA, it will find other things that it can interpret as more addresses, but which are not. These phony addresses may duplicate the real addresses of pages belonging to other files, thus causing MDAs. After IDAs are taken care of, files with MDAs may remain. Deleting the single file with the most MDAs may take care of the problem.

Once you have deleted the appropriate files, you should halt the system and bring it back up with disk recovery. If CHECKDISK again finds errors, you must again correct them, bring the system down, and then back up. If the fourth time CHECKDISK is run it still finds errors, notify an OAD systems programmer. Remember that once the system does come up successfully, the owners of the files must be notified about all files deleted.

13.3 Running CHECKDISK Yourself from EXEC

Introduction

CHECKDISK is a program that checks the address system and page allocation of the disk. CHECKDISK usually runs automatically as the system comes up. However, there may be occasions, for example after a standalone recovery, when you need to run CHECKDISK yourself. This section documents that procedure. What CHECKDISK does is explained in section 13.2, Correcting Errors Found by CHECKDISK.

Summary

1) At the EXEC "@", type "<system>checkdisk<ESC><CR>".

2) When CHECKDISK asks, "Do you want to run in multiple fork mode?", type "Y", for yes". All answers to CHECKDISK's question must be capitalized. Do not type more than a single letter, since CHECKDISK will take any excess letters as answers to following questions.

3) When CHECKDISK asks, "Do you want to run backwards?", type "N", for no.

4) To the question: "Rebuild the bit table?", type "N".

5) To the question: "Scan for disk addresses?", type "N".

6) CHECKDISK will now check the disk for bad files. For instructions on how to deal with bad files, see section 13.2.

Discussion

You invoke CHECKDISK by typing "<system>checkdisk<ESC><CR>". Once CHECKDISK is loaded, it will ask you a series of questions to determine how the disk should be checked and how much information about its status you want to get and store. When CHECKDISK runs automatically, these options are already specified; however, when you run CHECKDISK manually, you must specify them yourself.

The first question CHECKDISK will ask is, "Do you want to run in multiple fork mode?". This means, "Do you want to fire up a different fork of EXEC to run CHECKDISK separately for each disk?". The standard answer here is "Y", for "yes", since running CHECKDISK simultaneously on all the disks is faster than going through the disks one at a time. Note that you should type only the first letter of your answers to CHECKDISK's questions and that this letter must be capitalized. This is important. CHECKDISK cannot recognize lowercase letters. Moreover, if you type more than one letter, CHECKDISK will read the second and following letters as answers to later questions. This can cause a lot of problems.

Once CHECKDISK knows how many forks you want, it will ask if you want it to run backwards and check the disk from the last file to the first. The standard answer here is "N", for "no". CHECKDISK will then ask, "Rebuild the bit table?". Again, answer "N". The bit table is used to keep track of which pages on the disk have been used and which are free. However, the bit table is not updated after every process that frees pages in the disk. When you delete a bad file, for instance, the bit table will still mark as taken the pages that you have freed. Thus, it is a good idea to rebuild the bit table occasionally; otherwise, the whole disk could eventually be marked as taken, when parts of it were actually free. But rebuilding the bit table is too time consuming a process to do when you are bringing the system up from a presumably unscheduled crash.

The CHECKDISK will then ask if it should scan for disk addresses. CHECKDISK wants to know if you want the names of the files that are actually associated with all the bad disk addresses. Since this information is useful only to systems programmers, answer "N".

CHECKDISK will now check the disk and print a list of bad files and their errors. For instructions on how to deal with bad files, see section 13.2, Correcting Problems Found by CHECKDISK.

13.4 Deleting and Expunging Files

Summary

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) Connect to the directory that contains the file by typing "cd<SP>directoryname<CR>", where directoryname stands for the name of the directory you need.

3) Type "del<SP>filename<CR>". Make sure you type the entire file name including extensions. Do not use <ESC> to fill out the name -- the file may not be recognized correctly. Procede all unusual characters in the file name, for example @, with <CTRL-V>.

4) If the system tells you the file is perpetual, type "not<SP>perp<SP>filename" and then delete the file.

5) Type "exp<ESC><CR>".

6) Remember to connect back to directory "oper" when you finish deleting files by typing "cd<SP>oper<CR>".

NOTE: Always send a message to any user whose files you have deleted.

13.5 Halting the System

Introduction

There are two ways of halting the System XXV both halt the system immediately and for no designated length of time. They are used when you have encountered some problem during a crash recovery and want to bring the system down so that you can start again in the normal way. Method A is designed to halt a system that is running and will respond to commands given from the operator's terminal. This is probably the procedure you will most often use. You would use Method A, for example, to halt the system after fixing file problems found by CHECKDISK. Whenever Method A does not work because the system is hung or for some reason does not respond to the operator's terminal, you should resort to Method B. After halting the system with either of these methods, you may use whatever recovery procedure seems appropriate bring it back up.

Method A. Halting a Running System from the Operator's Terminal

This procedure has two steps. First, you need to get into MINI-EXEC, and then you need to halt the system. If you are already in MINI-EXEC when you decide to halt the system, start this procedure on step 4; if you are not, start at step 1. The way to tell if you are in MINI-EXEC is to look at the prompt. If it is a period (.), you are in MINI-EXEC; if it is anything else, you are not.

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) Type "quit<CR>"

3) When the system asks, "Do you really want to go into AUGUST monitor? (Confirm)", type "<CR>".

5) At the period (.) prompt, type "h".

6) The system will echo, "HALT TENEX".

7) Type ".".

8) The system will halt.

Method B. Halting the System from the Control Panel

1) Put address switch 31 on (up).

2) Put data switch 2 on.

3) Put CONSOLE DEPOSIT THIS on.

4) Put data switch 2 off.

5) Put data switch 0 on.

6) Put CONSOLE DEPOSIT THIS on.

7) When activity (the flickering of the lights, etc.) stops; the system has halted.

13.6 Changing the Date and Time

Summary

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) When you see the prompt "!", type "<CTRL-E>set<SP>DD-MON-YY<SP>HH:MM<CR>"; that is, two numbers for the day, a dash, the first three letters of the month, a dash, and then two numbers for the year. Follow this with a space, then give the time on 24 hour basis, and end with <CR> You must type the entire date and time to reset any part of it.

3) Type a confirming <CR>.

4) At the EXEC "@", type "day<CR>" to check the new date and time.

13.7 Connecting to and Disconnecting from the Micronode TYMBASE

Introduction

The two sets of procedures documented below allow you control whether or not the system will communicate the micronode TYMBASE. The ability to control the system's interaction with the micronode is useful when some micronode error is causing system problems or when the micronode is down and the system should not try to connect to it. Each set of procedures allows you to do the same things: Turn the micronode connection off, which causes the system to ignore the micronode; and turn the micronode connection on, which tells the system to synchronize with the micronode. Method A and Method B differ in where you give the controling commands. Method A uses commands given in EXEC. In Method B, in the other hand, you use EDDT. Method B should be used only during crash recovery, when you must control how the system interacts with the micronode as it comes up. In all other cases, control interaction from the EXEC with Method A.

Method A: Controling interaction from EXEC

To turn off the micronode connection

1) At the EXEC "@", type "<CTRL-E>tymnet<SP>off<CR>".

To turn on the micronode connection

1) At the EXEC "@", type "<CTRL-E>tymnet<SP>on<CR>".

Method B: Controling interaction from EDDT

To turn off the micronode connection

1) In EDDT, type "tymflg/".

2) After the system prints a value, type "0<CR>".

To turn on the micronode connection

1) In EDDT, type "tymflg/".

2) After the system prints a value, type "-1<CR>".

APPENDIX

This section is designed for quick reference: use it when you need to look up a certain step in a procedure, cannot remember exactly what order to do things, and so forth. No explanations of when to use these procedures, discussions of what they do, or suggestions about what to do if things go wrong are included here. For this type of information go to the first part of this manual where the procedures outlined in this section are discussed in greater length. All sections references in this appendix are also to earlier sections in this document.

WHAT TO DO IF THE SYSTEM IS HUNG

1) Put address switch 31 on (up).

2) Put data switch 2 on.

3) Put CONSOLE DEPOSIT THIS momentarily on.

4) Put data switch 0 on.

5) Put CONSOLE DEPOSIT THIS momentarily on.

6) Wait until activity (the flickering of the lights, etc.) stops.

7) Bring the system up with the disk recovery procedure.

DISK RECOVERY

1) Record the BUGHLT number and error lights.

2) Type "dskrld<ESC>g". The reponse should be "reloading from disk". If you never get this message, begin a tape recovery.

3) When the system says, "BOOT FROM DISK PACK # [CR FOR ANY]", type <CR>.

4) When the operator's terminal says "EDDT", type "start<ESC>g".

5) If CHECKDISK runs successfully, the system will announce "August in operation" and ask for the date and time. Enter these in the form DD-MON-YY<SP>HH:MM and follow with <CR>.

6) At the @ prompt, log in by typing "oper<SP>password<SP><CR>", where password stands for your password.

7) Type "ena<CR>".

8) Type "ref<SP>a<CR>".

AUTOMATIC RECOVERY

If a System XXV set for automatic recovery comes up successfully, you do not need to do anything until you log in as an operator.

1) At the @ prompt, log in by typing "oper<SP>password<SP><CR>", where password stands for your password.

2) Type "ena<CR>".

3) Type "ref<SP>a<CR>".

TAPE RECOVERY

1) Record the BUGHLT number and error lights.

2) If the power has gone off, you must reload the microcode as documented in steps 3 through 12. If the power has not gone off, skip to step 13.

3) Mount the microcode tape on the tape drive.

4) Put all switches on the control panel off (down).

5) Put address switch 32 on (up).

6) Put MICRO PROCESSOR STOP on.

7) Put MICRO PROCESSOR MIPC on.

8) Put MICRO PROCESSOR CLR momentarily on.

9) Put MICRO PROCESSOR CONT momentarily on.

10) Put MICRO PROCESSOR MIPC off.

11) Put MICRO PROCESSOR STOP off.

12) Put MICRO PROCESSOR CONT momentarily on. The tape should spin and then stop. Remove the microcode tape from the tape drive.

13) You are now ready to read in the new monitor. Mount the monitor tape on the tape drive. [Start here if you do not want to load the microcode.]

14) Put all switches on the control panel off.

15) Put address switches 24 and 26 on.

16) Put MICRO PROCESSOR STOP on.

17) Put MICRO PROCESSOR MIPC on.

18) Put MICRO PROCESSOR CLR momentarily on.

19) Put MICRO PROCESSOR CONT momentarily on.

20) Put MICRO PROCESSOR MIPC off.

21) Put MICRO PROCESSOR STOP off.

22) Momentarily put MICRO PROCESSOR CONT on. The tape should spin and then stop.

23) Put address switches 24 and 26 off.

24) Put address switches 29 and 30 on.

25) Momentarily put CONSOLE START on twice.

26) When the operator's terminal says "EDDT", type "start<ESC>g".

27) Remove the monitor tape from the tape drive.

28) After the system reports the size of the memory, put MI PAR ERR STOP and MEM PAR ERR STOP on.

29) If CHECKDISK runs successfully, the system will announce "August in operation" and ask for the date and time. Enter these in the form DD-MON-YY<SP>HH:MM and follow with <CR>.

30) At the @ prompt, log in by typing "oper<SP>password<SP><CR>", where password stands for your password.

31) Type "ena<CR>".

32) Type "ref<SP>a<CR>".

STANDALONE RECOVERY

1) Record the BUGHLT number and error lights.

2) Follow the procedure for tape recovery (section 8) from step 13 through step 25. If you suspect there has been a power failure, do step 3 through 25 of the tape recovery procedure.

3) When the operator's terminal says "EDDT", type "dbugsw/".

4) Type "2<CR>".

5) Type "start<ESC>g". The monitor tape should spin.

6) The system will request the date and time. Enter these in the form DD-MON-YY<SP>HH:MM and follow with <CR>.

7) You will automatically be logged in as "system", but not enabled.

DISK REBUILD STRATEGY OR TOTAL CATASTROPHE

WARNING: Never attempt this without specific instructions from a manager or an operating systems programmer.

1) Check with Tymshare Maintenance to make sure the hardware is good.

2) Follow the tape recovery procedure from steps 13 through 25. If there has been a power failure, do steps 3 through 25.

3) When the operator's terminal says "EDDT", type "dbugsw/".

4) Type "2<CR>".

5) Type "syslod<ESC>g".

6) When the system asks, "Do you really want to clobber the disk by reinitializing?", type "y<CR>".

7) Load the DLUSER tape on the tape drive.

8) Type "l", for "load". When the system asks, "Load from magtape MTAN:", type "mta0:<CR>" ("0" here is zero). Confirm this with another <CR>.

9) When the system asks, "File Number?", type "0<CR>".

10) When the system has read the DLUSER file and prompts you with a period (.), type "s.".

11) When you see "Interrupt at nnn", where nnn is some number, type "l", for "load", then specify "mta0:<CR>", and confirm with another <CR>.

12) When the system asks, "File Number?", type "1<CR>".

13) At the period prompt, type "s.".

14) Mount the first Full Dump Tape. Make sure you load the Full Dump Tapes in numerical order.

15) To DUMPER's question, "DUMP, LOAD, CHECK, OR SINGLE?", type "l", for "load".

16) For the second question, "DO YOU WISH TO SUPERSEDE OLDER VERSIONS ALWAYS?", type "n".

17) For the question, "SPECIFIC USERS?", type "n".

18) To answer, "INTO SAME DIRECTORIES?", type "y".

19) When requested, "TYPE MAG TAPE UNIT NUMBER", type "0" (zero).

20) DUMPER will now read the tape; when it is finished, it will print, "MOUNT NEXT TAPE, IF ANY. TYPE C, WHEN READY, N, IF NO MORE". Mount the next tape and type "c".

21) When DUMPER asks for the mag tape unit number; type "0".

22) Continue mounting and loading Full Dump Tapes, typing "c" to continue and then giving 0 (zero) for mag tape unit number, until all the tapes have been read.

23) When you have finished loading the Full Dump Tapes, begin loading the Incremental Dump Tapes. Be sure that you load the Incremental Dump Tapes in chronological order, beginning with the one made right after the Full Dump and ending with the most recent.

24) After the system has read the last Incremental Dump Tape, when DUMPER says, "MOUNT NEXT TAPE, IF ANY. TYPE C, WHEN READY, N, IF NO MORE", type "n".

25) When you see an interrupt message followed by a period, halt the system, and begin a disk recovery. (To halt the system use Method A of section 13.5, Halting the System.)

CHANGING RECOVERY SWITCH SETTINGS

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) At the "!" prompt, type "mddt<CR>".

3) Type "switchname/", where switchname stands for the name of the switch you want to change.

4) Type "switchvalue<CR>", where switchvalue stands for the new value the switch should have: 0, 1, or 2.

5) Type "<CTRL-C>" to return to EXEC.

CORRECTING PROBLEMS FOUND BY CHECKDISK

1) If only one file has errors, delete and expunge that file. Be sure to type the entire file name.

2) If more than one file is involved, delete and expunge all files with IDAs. Do not delete the files with MDAs at this point.

3) Halt the system with Method A of section 13.5 and bring it up with disk recovery. If CHECKDISK again finds files with IDAs, repeat this procedure.

4) Once no files have IDAs, if one or more files have MDAs, delete and expunge the file with the largest number of MDAs, halt the system, and bring it up with disk recovery. Do this three times. If you then still have more than 20 files with MDAs, call an operating systems programmer.

NOTE: Keep a list of the files you delete and expunge, and restore them after the system comes up. Always send messages to all users whose files have been deleted and restored.

RUNNING CHECKDISK YOURSELF FROM EXEC

1) At the EXEC "@", type "<system>checkdisk<ESC><CR>".

2) When CHECKDISK asks, "Do you want to run in multiple fork mode?", type "Y", for yes". All answers to CHECKDISK's question must be capitalized and one letter.

3) To the question, "Do you want to run backwards?", type "N", for no.

4) To the question: "Rebuild the bit table?", type "N".

5) To the question: "Scan for disk addresses?", type "N".

6) CHECKDISK will check the disk for bad files. For instructions on how to deal with bad files, see section 13.2.

DELETING AND EXPUNGING FILES

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) Connect to the directory that contains the file by typing "cd<SP>directoryname<CR>", where directoryname stands for the name of the directory you need.

3) Type "del<SP>filename<CR>". Make sure you type the entire file name including extensions. Do not use <ESC> to fill out the name -- the file may not be recognized correctly. Procede all unusual characters in the file name, for example @, with <CTRL-V>.

4) If the system tells you the file is perpetual, type "not<SP>perp<SP>filename" and then delete the file.

5) Type "exp<ESC><CR>".

6) When you finish deleting files, type "cd<SP>oper<CR>".

NOTE: Always send a message to any user whose files you have deleted.

HALTING THE SYSTEM

Method A. Halting a Running System from the Operator's Terminal

This procedure has two steps. First, you need to get into MINI-EXEC, and then you need to halt the system. If you are already in MINI-EXEC when you decide to halt the system, start this procedure on step 4; if you are not, start at step 1. The way to tell if you are in MINI-EXEC is to look at the prompt. If it is a period (.), you are in MINI-EXEC; if it is anything else, you are not.

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) Type "quit<CR>"

3) When the system asks, "Do you really want to go into AUGUST monitor? (Confirm)", type "<CR>".

5) At the period (.) prompt, type "h".

6) The system will echo, "HALT TENEX".

7) Type ".".

8) The system will halt.

Method B. Halting the System from the Control Panel

1) Put address switch 31 on (up).

2) Put data switch 2 on.

3) Put CONSOLE DEPOSIT THIS on.

4) Put data switch 0 on.

5) Put CONSOLE DEPOSIT THIS on.

6) When activity (the flickering of the lights, etc.) stops; the system has halted.

CHANGING THE DATE AND TIME

1) If your prompt is not an "!", type "ena<CR>" at the EXEC "@".

2) When you see the prompt "!", type "<CTRL-E>set<SP>DD-MON-YY<SP>HH:MM<CR>"; that is, two numbers for the day, a dash, the first three letters of the month, a dash, and then two numbers for the year. Follow this with a space and then give the time on 24 hour basis. You must type the entire date and time to reset any part of it.

3) Type a confirming <CR>.

4) At the EXEC "@", type "day<CR>" to check the new date and time.

CONNECTING TO AND DISCONNECTING FROM MICRONODE TYMBASE

Method A: Controling interaction from EXEC

To turn off the micronode connection

1) At the EXEC "@", type "<CTRL-E>tymnet<SP>off<CR>".

To turn on the micronode connection

1) At the EXEC "@", type "<CTRL-E>tymnet<SP>on<CR>".

Method B: Controling interaction from EDDT

To turn off the micronode connection

1) In EDDT, type "tymflg/".

2) After the system prints a value, type "0<CR>".

To turn on the micronode connection

1) In EDDT, type "tymflg/".

2) After the system prints a value, type "-1<CR>".

Price $3.00

COMMANDS STRUCTURES
AND THE PROCESS COMMAND

March 1981
Journal Number 75526

Office Automation Division
Tymshare, Inc.
20705 Valley Green Drive
Cupertino, California 95014

TABLE OF CONTENTS

INTRODUCTION

The Process command in the Base subsystem can speed up many time-consuming, repetitive, or routine tasks by allowing you to automatically execute a series of commands. This is done by writing the commands as text into any convenient place, like your initial file. These commands can then be processed by using the Process command and marking or addressing the stored text. The structure containing the text is called a "commands structure".

A commands structure is usually written as a branch, and so it is often referred to as a "commands branch". The Process command, however, can be used to execute commands in a single statement, a branch, a group, or a plex.

Writing commands structures is analogous to writing what in other systems are called "Commands Files" or "macros" without arguments.

This document describes how to write a commands structure and how to use the Process command. It is written primarily for users of AUGMENT in display mode; however, those who use a typewriter terminal will also find this document useful. Differences between display and typewriter mode, where they exist, are explained. This document offers some general comments about processing commands structures, suggestions on testing them, warnings about editing with them, and annotated examples of commands structures written by AUGMENT users.

SOME THINGS TO KNOW BEFORE WRITING A COMMANDS STRUCTURE

A commands structure may contain AUGMENT commands only. You cannot use Executive commands in an AUGMENT commands structure, but you can go from the Base subsystem to other subsystems.

When writing a commands structure, you should use complete command words, not abbreviations. This ensures that the command words will be recognized by AUGMENT, and also allows you and others to read your commands structure easily. Words in commands structures can be written in any combination of lowercase and uppercase letters. Some users, however, prefer to capitalize the first letter of command words so they can read the commands more easily. In any case, all command words should be followed by one space (or <ESC>) to be recognized by AUGMENT.

The text of commands structures also includes the special characters you need to supply in the command. When writing the text of commands structures with the Insert Statement command, you can enter characters such as <OK> (or <CTRL-D>), <NULL> (<CTRL-N>), or any control character by first typing <LIT> (<CTRL-V>). When <LIT> is typed before a special character, it prevents AUGMENT from interpreting the character for immediate execution, thus allowing you to include it in the text of your commands structure; in other words, it causes AUGMENT to take the character literally. Once entered in the text, the special character will be displayed as a series of uppercase letters surrounded by angle brackets. (<LIT> will not appear because it was not entered in the text.)

For example, to include <OK> in a commands structure, you would first type <LIT> and then <OK>; only "<OK>" will appear as part of your command. Special characters that appear in this way in commands structures are single characters. "<OK>" is one character, not four characters, and can be inserted, transposed, moved, or deleted as a single character. Writing out "<OK>" using angle brackets and uppercase letters will not work because AUGMENT will not read that text as a special character.

When you write commands structures, special characters appear differently in your command window than in the text displayed in your file window. For example, when you type <LIT><OK>, only "!" appears in your command window, but "<OK>" appears in the text of your structure. Note that if you print a commands structure, the special characters will not be printed.

WRITING AND PROCESSING A COMMANDS BRANCH

This section describes how to set up and process a commands structure in the form of a branch. The first statement in the branch serves as the name of the branch, and the substatements contain the commands to process. The same general procedure can be used to create a commands structure in the form of a statement, group, or plex, as described later.

1. Insert in your file a statement naming your commands branch.

   This first statement should be short, preferably one word, possibly abbreviated, and easy to remember. It should be within parentheses and should include no punctuation. If you use more than one word to avoid confusion with the names of other commands branches in your file, run the words together or separate them with dashes (-). For example, the first statement might look like this:

   ```
   (dirin)
   ```

   Include no other information besides the name in the first statement.

2. Specify that parentheses are the name delimiters of the branch.

   Parentheses are conventionally used as the name delimiters of a commands branch. To specify that they be recognized as the name delimiters, use the Set Name (delimiters) command. Its form, or "syntax", is:

```
Set Name (delimiters in) Branch (at) LOCATION (left delimiter
to be) CONTENT (right delimiter to be) CONTENT
```

To indicate LOCATION, mark the first statement or type its address followed by <OK>. For the first CONTENT, type a left parenthesis followed by <OK>; for the second, type a right parenthesis followed by <OK>.

Note that the effect of these first two steps is to make the first statement the name of the branch.

3. Insert command statements into the branch in a logical order to complete the task you want to perform.

While more than one command can be included in each statement, to avoid confusion you should insert a separate statement for each step in your commands branch. If a command is particularly long, you can continue it to the next statement. The statements containing commands should be one level down from the first statement. A space (or <ESC>) must follow each command word. Remember to type <LIT> before special characters (see example below). The order of the commands and the syntax of each command must be absolutely correct for the commands branch to be processed correctly. An incorrect command order or syntax can cause unexpected problems.

NOTE: Do not include noise words in command statements. Include only the parts of the command that you must specify when you give the command.

For example, if you enter

```
jump<SP>link<SP>index<LIT><OK>
```

the statement will look like this in your file window:

```
jump link index<OK>
```

4. Test the commands branch before running it.

Testing allows you to make sure that the command syntax and the order in which the commands are given are correct. The testing procedure is described later under TESTING A COMMANDS STRUCTURE.

5. To run the process, use the Process command.

If addresses in the structure refer to an unspecified current file, make sure you are in the file you want the commands to act upon before you give the Process command. The form of this command that processes a branch is:

```
Process (commands from) Branch (at) LOCATION
```

For LOCATION, type the name you gave the first statement, without the parentheses, followed by <OK>. If the structure is in another file, precede the statement name with the file name and a comma. Remember that the Process command may also be used for a commands structure in the form of a statement, group, or plex. Processing these structures is described later.

Normally a process will stop automatically when all of the commands are executed. There are, however, two ways of stopping a process before all commands in the structure are executed. Typing <CTRL-O> while a process is running stops it; typing <CTRL-C> stops it and returns you to the Executive.

AN EXAMPLE OF A COMMANDS BRANCH

The following is an example of a commands branch that copies a list of the files in the current directory into a branch named "index", replacing the previous list, if any.

```
(dirin)

   insert statement index<OK>d<OK>temporary<OK>

   delete plex index.d<OK><OK>

   copy directory <OK>index<OK>d<OK>no version <OK><OK>

   jump link index<OK>
```

The name delimiters of the statement "(dirin)" have been set to parentheses. The first step in the process inserts a "temporary" statement one level below the statement named "index" in the current file. This is done because there must be substructure under the "index" statement for the next command, Delete Plex, to find something at that level to delete; otherwise, the entire contents of the file may be deleted. The second step deletes the plex one level below the "index" statement, including the temporary statement plus any substructure already there. The third step writes the directory list to follow a level below the "index" statement without indicating version numbers. The fourth step jumps to the "index" statement.

NOTE: The first step of the example above, the insertion of a temporary statement below the "index" statement, is a precaution that should be taken for any command statement that locates a structure with relative addressing (such as the address "index.d") and deletes that structure. If the structure intended for deletion is not there, the address points one level higher. Inserting a "temporary" or "dummy" statement ensures something will be there to delete, thus avoiding damage to a file.

You can adapt the commands branch shown in the example above for your initial file. Before the commands branch will work, there must be a statement named "index" in the file. You will have to be in your initial file when you run the branch.

USING THE PROCESS COMMAND ON OTHER STRUCTURES

While a named commands branch is the most convenient and most widely used commands structure, there are certain times when it is useful to process commands in a statement, group, or plex.

The Process Statement command allows you to process a single command statement, either by marking the statement or typing its address. This is useful when you want to execute a single command, and is especially useful in writing and testing longer commands structures. The form of this command is:

```
   Process (commands from) Statement (at) LOCATION
```

The Process Group command is helpful when you want to process only a portion of a commands structure. When you use this command, only the consecutive group of command statements you mark or address will be processed. The form of this command is:

```
   Process (commands from) Group (at) LOCATION (through) LOCATION
```

The Process Plex command is useful because an entire plex containing command statements can be processed by marking or typing the address of any statement in the plex. This is just like a branch without the first statement; it refers to all the statements at that level with the same upstatement. The form of this command is:

```
   Process (commands from) Plex (at) LOCATION
```

NOTE: The name delimiters of any commands structure must be changed from the default before it will work. The reason for this is that the Process command skips over statement names. The Set Name (delimiters) command also can be used to change name delimiters in a statement, group, or plex. If the statement, group, or plex is in the substructure of an existing commands branch, it is not necessary to specifically change the name delimiters.

TESTING A COMMANDS STRUCTURE

Testing a commands structure is important because neither the Process command nor commands structures contain logical error-checking capabilities, so they are not able to determine whether unwanted changes are being made in a file. Testing will ensure that the commands structure does just what it is intended to do, and will allow you to make sure that command syntax and the order of commands are correct. Individual command statements, part of the structure, or the entire structure can be tested on an AUGMENT display terminal in either display or typewriter mode, or on a typewriter terminal in typewriter mode. If there is a mistake in the text of a commands structure, any one of a number of things might happen:

The process might do something other than what was intended.

The process might stop or an error message may appear in your status window.

AUGMENT might try to process an incorrectly written command statement by taking the first characters it can find to execute a command, giving no error message.

In any case, it is important to remember that while a commands structure is being tested, the commands are actually being carried out.

When a commands structure runs in display mode, the commands briefly appear in your command window as they are executed, but they usually appear too briefly for you to read them, so mistakes are hard to see. The advantage of testing part or all of a structure in typewriter mode or at a typewriter terminal is that it allows you to identify exactly where any problems with command syntax may exist. In typewriter mode, the text of a command statement will "scroll" out from the bottom of your screen; it will not disappear after the command is executed as it does in display mode. On a typewriter terminal, you will have a printed version which can be checked for accuracy. If there is a mistake it will be obvious, in a way described below.

If you are not using AUGMENT at a typewriter terminal, you may set your display to typewriter mode with the Set Terminal command, as follows:

```
Set Terminal (mode to be) Typewriter <OK>
```

To test individual statements of a commands structure in typewriter mode, use the Process Statement command:

```
Process (commands from) Statement (at) LOCATION
```

In typewriter mode, you should specify LOCATION by typing the statement number of the command statement you wish to test. If you want to test a statement that is in another file, you must type the file name, a comma, and the statement number. For example, suppose you wanted to test this statement:

```
jump link index<OK>
```

For LOCATION in the Process Statement command, you would type the statement number of the statement. This would appear on your screen:

```
   Jump  (to) Link  index!
```

If, however, there is something wrong with the command statement, then something like this may appear on your screen:

```
   \ Ln\nn\nk  I\IN\NK ...
```

If a character or characters repeat after a slash mark, it means that they could not be logically processed at that point in the command, and AUGMENT is trying to "backspace" them. When this happens, the bell will ring. Backspacing can be caused if you misspelled a command word, omitted the space after a command word, failed to type <LIT> before a special character, failed to change the name delimiters, or gave incorrect command syntax. The first instance of backspacing appears where the first unusable character appears in your command statement; that character is a space in the example shown.

To correct a faulty command statement, set your terminal to display mode again using the Set Terminal command:

```
   Set Terminal (mode to be) Display <OK>
```

Once in display mode, you can check the command statement for omitted or extra spaces between command words, misspelled command words, or faulty command syntax. You can check name delimiters by using the Show Name (delimiters for statement) command.

Although AUGMENT clearly signals a mistake in the text of command statements, there is no similar signal if the order of commands is inappropriate for the task you wish your commands structure to perform. By trying out each command statement with the Process Statement command, as discussed above, and looking at the results, you can determine whether the order of commands is correct.

Note that there are differences between typewriter mode and display mode in the way commands are given, and there are some Base commands that cannot be given in typewriter mode. If the commands structure includes commands that work only in display mode, you cannot test them in typewriter mode as suggested above. Instead you must carefully watch your command window for errors and question marks while processing each statement in display mode.

If you have acquired a commands structure that has been used by someone else for some time, you still should test it to make sure you have all your files and statements set up as the process requires, and to make sure it was written for the current version of AUGMENT.

GENERAL SUGGESTIONS

You should use branches for your commands structures whenever possible. These can be inserted in your initial file for convenience. If you wish to process a structure in another file, remember that the file name must be specified.

When deciding exactly what to include in your commands structure, give each command at your display terminal and jot down the steps on a piece of paper, or enter the commands at a typewriter terminal so you will have a printed copy of what you did.

You can have a commands branch start automatically every time you enter AUGMENT in display mode as follows:

```
   Set Useroptions startup commands branch.display<OK> (specify
   value) CONTENT
```

To have a commands branch start automatically in typewriter mode, type ".typewriter" instead of ".display"; otherwise, the syntax is the same. For CONTENT, mark or type a link to your branch. For example, you could type the file name, a comma, and the statement name of your branch, ending with <OK>. You can have one commands branch in display mode and one commands branch in typewriter mode specified at any one time. You can cancel the automatic processing, starting with the next AUGMENT session, as follows:

```
   Reset Useroptions startup commands branch.display<OK> <OK>
```

Similarly, by typing ".typewriter" instead of ".display", you can cancel automatic processing of a commands structure that runs in typewriter mode.

You should check your commands structures once in a while and keep them current with changes in AUGMENT.

Include an Update command early in the process so that if unwanted editing occurs you can delete modifications and recover.

If the file space allocation is likely to be near the limit and there are some large files that the process updates, include an Expunge Directory command to minimize the chance of exceeding your allocation.

Comments should be used where the logic of the commands structure is obscure. This makes it easier to come back to the structure later and understand what it is intended to do. Comments can be included at the end of command statements or as separate statements by placing a semicolon (;) and a space at the start of the comment and a literal <OK> at the end. A command statement with a comment might look like this:

```
   jump link index.d<OK>; jumps one level down from the statement
   "index"<OK>
```

A comment is displayed briefly on your screen or printed on your typewriter terminal, but it is not interpreted as a command. Note that a commands structure containing comments takes somewhat longer to process.

If you plan to run a commands structure in both display mode and typewriter mode (or run it in the former and test it in the latter), there are a few differences between the two modes that you should be aware of.

> Besides differences in the command words that are allowed, there are differences in addressing. In typewriter mode, you can use <OK> to specify the current location whenever an address is expected, whereas in display mode, <OK> is useless for addressing, since it acts as a <MARK> of any randomly located statement on the screen. To specify the current location in either mode, you can use ".<OK>".
>
> The current location itself may be different in typewriter mode than in display mode after certain commands, so be sure to test your commands carefully if they refer to the current location.

In some cases, it may be important to note which commands change the current location and which ones do not. For example, the address ".d" is relative to the current location, which may be changed by some editing commands. Also, the Move command, when used within a file, changes the current location to the new location of the statement.

WARNINGS ABOUT EDITING WITH COMMANDS STRUCTURES

If a file is left unchanged or unexamined for a long period of time, it may be archived. So, when addressing a file or statements within a file with a commands structure, it is wise to make sure that the file has not been removed from your directory by being archived. If a commands structure contains a command to edit a file that has been archived or deleted from a directory, it will work instead on the current file. If a commands structure jumps between files and one or more of these files are no longer in the directory, changes may be made in the wrong files. This can cause unexpected and unwanted changes to your files.

If your file space allocation is exceeded while a commands structure runs, it may result in an uncompleted job. Include an Expunge Directory command early in a commands structure to minimize the chance of exceeding your allocation.

When you address a statement, plex, or other structure using an infile address element that specifies structural position, such as .s (successor), .d (down), and .n (next), make sure that the intended statement will be there in all cases. In cases where a commands structure may find nothing to work on, you must have your commands structure insert a "dummy" or "temporary" statement, as in our initial example. Also, remember that infile address elements of this type change the current location to the structure specified by the address element.

A commands structure can be made to perform repetitive editing tasks, such as deleting a certain character from every statement in a file, by including a command in the structure to process part of itself. Caution should be used with such indefinitely self-processing commands structures, however, because when the task is completed, the structure will still continue processing. In such a case it is necessary to use <CTRL-O> or <CTRL-C> to stop the process. A method of making a commands structure process itself a specified number of times is offered below under ANNOTATED EXAMPLES OF COMMANDS STRUCTURES.

When a commands structure addresses statements by number, consider whether the statement numbers may change as editing occurs. In these cases, SIDs may be more dependable.

At the end of a process a <CD> is generated, so you cannot leave a command unfinished in a process and expect to finish it manually.

ANNOTATED EXAMPLES OF COMMANDS STRUCTURES

Each of the following commands structures is preceded by a title and an introduction. Each command statement is followed by a brief explanation in brackets. Comments in brackets are used here for explanation only and cannot be included as comments in commands structures. The spaces that appear in the command statements are required, and <LIT>s typed before control characters are not shown.

NOTE: These examples have been submitted by their authors and are included here as illustrations of useful or novel applications of commands structures. You are welcome to adapt any of these for your own files, if you wish; however, do so at your own risk.

A Commands Branch to Record Number of Accesses, Creation Dates, Size, and Last Writer for a File

> This commands branch is used to determine, for a specified file, the number of times it has been accessed, the creation date of the current version, the creation date of the original version, the size, and the last writer. It copies this information into your file below a specified statement, replacing the old information, if any. DIRECTORYNAME, FILENAME, and STATEMENTNAME must be replaced with specific names before the branch is processed.

```
(fileshow)

   insert statement STATEMENTNAME<OK>d<OK>dummy<OK>
```

[Inserts a "dummy" statement one level below a specified statement. This is to ensure there is substructure when the plex is deleted.]

```
   delete plex STATEMENTNAME.d<OK><OK>
```

[Deletes the plex one level down from the specified statement. This step deletes both the dummy statement and any information left the last time the branch was run.]

```
   copy directory DIRECTORYNAME<OK>STATEMENTNAME<OK>d<OK>for
   FILENAME<OK>number accesses date creation date first size
   last <OK><OK>
```

[Copies one level down from the specified statement: the file name of the most recent version, the number of accesses, the creation date of the current version, the creation date of the original version, the size, and the last writer.]

```
   jump link STATEMENTNAME<OK>
```

[Jumps to the branch containing the information, displaying it at the top of your display window.]

A Commands Branch to Return to the Last Modification of a File

This commands branch will return you to the point in a file where the last modification was made. It requires that before leaving the file you modified, you placed the text "***" in the last statement you changed. DIRECTORYNAME, PASSWORD, and FILENAME must first be replaced with the specific names.

```
(goback)

   connect directory DIRECTORYNAME<OK>PASSWORD<OK>
```

[Connects to the specified directory.]

```
   jump link FILENAME<ESC><OK>
```

[Jumps to the most current version of a specified file.]

```
   jump content first ***<OK>IGmwy<OK>
```

[Jumps to the statement that was last modified. Viewspecs are set to show all lines, all levels, blank lines between statements, and SIDs on the right.]

A Self-Modifying, Looping Commands Branch

This commands branch demonstrates two useful features of commands structures: They can be self-modifying, and they can work in a looping fashion. This example shows two versions of the same commands branch. The first shows each command statement and describes what it does, and the second shows how the branch should look in your file if you wish to try it. The two versions are offered because as the commands branch runs the structure of the branch itself changes. This commands branch will delete the last character from the first five statements after the origin statement of a specified file. To do this, the commands branch first adds statements to itself and then deletes them. This example is offered here to illustrate a useful feature that can be adapted to perform many editing tasks. FILENAME must be replaced by the name of the file you want to edit. This commands branch works only in display mode.

```
(jump-del)

   insert statement loop.dn<OK>delete statement
   loop.dt<LIT><OK><LIT><OK><OK>
```

[This statement inserts the command statement "delete statement loop.dt<OK><OK>" one level down from the next statement following the "(loop)" statement below. The <LIT>s are included by typing <LIT> twice.]

```
   insert statement loop.d2n<OK>process branch loop<LIT><OK><OK>
```

[This statement inserts the statement that will cause the branch to loop. The commands branch is set up to insert these first two statements into itself because a later command statement will delete them. Having them inserted automatically every time the branch is run saves time and typing.]

```
   jump link FILENAME,<OK>
```

[Jumps to the origin statement of the specified file.]

```
   process branch loop<OK>
```

[Starts the commands branch at "(loop)" below.]

```
   (loop)
```

[Names the second commands branch.]

```
      <LF>
```

[Jumps to the next statement from the origin, then jumps to the next statement from the current one as the commands branch loops. Since <LF> acts as a command word, it must be followed by a space. An alternative to <LF> in display mode is "jump link .n<OK>".]

```
delete character +e<OK><OK>
```

[Deletes the last character in the statement at the top of the file window.]

```
delete statement loop.dt<OK><OK>
```

[Deletes successive "x" or "dummy" statements, starting from the last one and working upward as the commands branch loops, then deletes the "process branch loop<OK>" statement. When it deletes that statement, the commands branch stops looping. Finally, this statement deletes itself.]

```
process branch loop<OK>
```

[Processes the "(loop)" branch again, jumping and deleting the last character in each consecutive statement in the file as the commands branch loops, until this statement is itself deleted.]

```
x
```

[Each "x" represents a "dummy" or temporary statement. The number of dummy statements determines how often the branch will loop. They do not represent commands. If no dummy statements are provided, the branch will automatically loop two times.]

```
x

x
```

To try this commands branch in a file of your own, set it up as follows:

```
(jump-del)
   insert statement loop.dn<OK>delete statement
   loop.dt<LIT><OK><LIT><OK><OK>
   insert statement loop.d2n<OK>process branch
   loop<LIT><OK><OK>
   jump link FILENAME,<OK>
   process branch loop<OK>
   (loop)
      <LF>
      delete character +e<OK><OK>
      x
      x
      x
```

With three "x" statements the commands branch will loop five times; you can insert as many "x" statements as you wish, remembering that the commands branch will loop twice if no "x" statements are provided.

A Commands Branch to Check Spelling and Record Errors

This commands branch will check an entire file or part of a file for misspelled words and record the misspellings in another file. The branch contains command statements to supplement the 42,000-word Spell subsystem dictionary with Output Processor directives and with a supplemental dictionary of your own, which may contain unusual spellings, proper names, or words not found in the Spell dictionary. Either of the statements containing a Supplement command can be omitted if you have no supplemental dictionary or if your file contains no Output Processor directives. Before processing the commands branch, you need a file in your directory to receive the recorded misspellings. Also, you must replace both instances of MISSPELLEDFILE with the name of the file that receives recorded mispellings, and you must replace DICTIONARYFILE with the name of the file containing your supplemental dictionary before processing the commands branch. You should be in the file you want to check, so remember that when you use the Process command, you must specify the directory name, file name, and name of the commands branch.

```
(spellcheck)

   execute programs delete all <OK>

      [Enters the Programs subsystem and deletes all programs in
      the buffer.]

   goto <OPT>spell<OK>

      [Enters the Spell subsystem.]

   supplement directives <OK>

      [Supplements the Spell subsystem dictionary with Output
      Processor directives.]

   supplement branch DIRECTORYNAME,DICTIONARYFILE,<OK>

      [Supplements the Spell subsystem dictionary with your own
      dictionary.]

   set mode recording DIRECTORYNAME,MISSPELLEDFILE,<OK>

      [Sets the Spell subsystem checking mode to record all
      misspelled words in the indicated file.]
```

```
check branch 0<OK><OK>

   [Checks spelling in the file you currently are in.
   Specifying the branch at 0 checks the entire file;
   however, any branch that you indicate in a file can be
   checked.]

quit <OK>

   [Leaves the Spell subsystem and returns to the Base
   subsystem when the checking is completed.]

jump link DIRECTORYNAME,MISSPELLEDFILE,:wz<OK>

   [Jumps to the file containing the recorded misspelled
   words with viewspecs set to show all lines and levels,
   with no blank lines between statements.]
```