```
***********************************************************
*                                                         *
*          USER'S PERSPECTIVE -- An Introduction          *
*                                                         *
*                           for                           *
*                                                         *
*  ZCPR3 -- Z80 Command Processor Replacement, Version 3  *
*                                                         *
***********************************************************
```

by

Richard Conn

User's Perspective
11 June 1984

ZCPR3 Version 3.0

THE USER’S PERSPECTIVE - AN INTRODUCTION TO ZCPR3

by Richard Conn

The ZCPR3 System is a collection of programs based around the ZCPR3 Command Processor. Forming an integrated system of tools, the ZCPR3 System offers a number of convenient and sometimes more user-friendly features to the CP/M 2.2 user. Maintaining CP/M 2.2 compatibility at all times (all known commercial CP/M 2.2 programs run under ZCPR3 without modification), the ZCPR3 System brings to its users a variety of tools which conceptually implement features found in other operating systems, including TOPS-20 (1), UNIX (2), NOS (3), MULTICS (4), and VMS (5), and tools which implement features unique to the ZCPR3 System (to my knowledge).

This Introduction is intended to outline some of the key features of the ZCPR3 System from the user’s perspective. Knowledge of CP/M 2.2 is assumed, and some experience with ZCPR2 is useful in order to understand the following presentation in detail. The major features of the ZCPR3 System which are described in this Introduction include:

```
     o Directories       o Wheel Users and Passwords
     o Command Lines     o Command Processing
     o Error Handlers        o Resident Command Packages
     o Aliases               o Flow Command Packages
     o "Secure" Systems      o ZEX Command Files
     o Shells            o Z3TCAP
         o Variable          o Screen-Oriented Terminal
         o MENU                  Configuration
         o VFILER
```

The following screen displays are intended to convey ideas only. These displays were generated while the ZCPR3 System was being developed, and the version numbers and operation of the ZCPR3 utilities as distributed differ from those shown in this document.

---- Trademarks ----

(1) TOPS-20 -	Digital Equipment Corporation
(2) UNIX -	Bell Laboratories
(3) NOS -	Control Data Corporation
(4) MULTICS -	Honeywell
(5) VMS -	Digital Equipment Corporation

1. Directories

Under ZCPR3, a logical disk can be thought of to contain two types of directories. One is the physical directory, which is usually located just after the system tracks on most floppies. The other is the logical directory, in which each file on a disk has a user number associated with it (from 0 to 31), and the combination of a disk and user number identifies uniquely the logical directory in which the file belongs. DDT.COM may be located on disk A, user 5, while two copies of ED.COM may be located on disk A, user 5 and disk A, user 0. The combination of the disk reference and user number identifies the logical directory which a file belongs in.

The logical directory is usually indicated as part of the prompt. In the examples below, the reader can see the logical directory referred to by its disk and user number and, in most cases, by a name associated with the disk and user number. The following examples illustrate the use of the DU (disk/user) form and the DIR (directory name) form to log into various user areas and directories.

```
A0:BASE>15:
A15:ROOT>4:
A4>b:
B4:WORK4>0:
B0:WORK1>a14:
A14>a0:
A0:BASE>root:
A15:ROOT>work2:
```

Commands may use either the DU or DIR form to reference the logical directories they are to act upon. Interpretation of the name of a directory is built into the ZCPR3 command processor itself, so every command can work with the DU and DIR forms with equal ease.

Commands like DBASE which don’t know about the DU or DIR forms will usually just pay attention to the disk referenced and not the user number. For commands like these, it is usually best to just employ the disk letter when referring to their arguments.

```
B1:WORK2>dir base:
 RHEX    .COM    2
            A0:BASE --     1 Files Using     2K (  206K Left)

B1:WORK2>base:
```

The PWD command displays the names and associated DU forms of all directories which currently have names assigned to them. Additionally, when using the DIR form to log into a directory (see PRIVATE below), a directory so named may have a password associated with it. If so, the user is prompted for this password and the command will fail if he does not provide the correct password.

```
A0:BASE>pwd
PWD, Version 1.0
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE     A 15: ROOT

B  0: WORK1       B  1: WORK2       B  2: WORK3       B  4: WORK4
B  5: TEXT        B  6: MAIL

A0:BASE>private:
PW? unknown

A0:BASE>private:
PW? mypass

A1:PRIVATE>dir
            A1:PRIVATE --     0 Files Using     0K (  206K Left)

A1:PRIVATE>base:

A0:BASE>dir private:
PW? mypass
            A1:PRIVATE --     0 Files Using     0K (  206K Left)
A0:BASE>dir a1:
            A1:PRIVATE --     0 Files Using     0K (  206K Left)
```

2. Wheel Users and Passwords

Password protection is common under ZCPR3. Several of the ZCPR3 utilities respond one way if the user is priveleged (a Wheel) or not priveleged. A user becomes priveleged by running the WHEEL command and giving the Wheel Password.

```
A0:BASE>mkdir
MKDIR, Version 3.0
 Permission to Run MKDIR Denied - Not Wheel

A0:BASE>pwd pass
PWD, Version 1.0
 Password Request Denied - Not Wheel
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE     A 15: ROOT

B  0: WORK1       B  1: WORK2       B  2: WORK3       B  4: WORK4
B  5: TEXT        B  6: MAIL

A0:BASE>wheel /s
WHEEL, Version 3.0
 Wheel Password?  Wheel Byte is ON

A0:BASE>pwd pass
PWD, Version 1.0
 DU : DIR Name - Password     DU : DIR Name - Password
----  --------   --------    ----  --------   --------
A  0: BASE     -             A  1: PRIVATE  - MYPASS
A 15: ROOT     -

B  0: WORK1    -             B  1: WORK2    -
B  2: WORK3    -             B  4: WORK4    -
B  5: TEXT     -             B  6: MAIL     -

A0:BASE>private:
PW? mypass

A1:PRIVATE>root:
```

```
A15:ROOT>mkdir sys.ndr
MKDIR, Version 3.0

MKDIR Command (? for Help)? C
** MKDIR Change Mode **
Directory Entry (?<RETURN> for Help)? a2:priv2
        Adding PRIV2    -- Password? mypass2
 10 Entries in Directory
Directory Entry (?<RETURN> for Help)?
 DU : DIR Name - Password     DU : DIR Name - Password
----  --------   --------    ----  --------   --------
A  0: BASE     -             A  1: PRIVATE  - MYPASS
A  2: PRIV2    - MYPASS2     A 15: ROOT     -

B  0: WORK1    -             B  1: WORK2    -
B  2: WORK3    -             B  4: WORK4    -
B  5: TEXT     -             B  6: MAIL     -

Directory Entry (?<RETURN> for Help)? x
MKDIR Command (? for Help)? X

Directory has changed since last Write
Do you want to write Directory to Disk (Y/N)? Y

Name of File (<RETURN> = A 15: SYS     .NDR)? special.ndr
Writing Directory to Disk ... Done
```

If a user knows the right passwords and has the proper Wheel privelege, he can radically change the directory structure, bringing new directories which were previously undefined into existence.

The ability to log into a directory can be controlled by the installer. At installation time, the ability to use DU and DIR forms to log into directories or reference directories can be established. On a more secure system, for example, the ability to use the DU form may be denied. Then, only directories defined by name may be accessed (DIR form), and, if these directories have passwords associated with them, the proper passwords must be given.

```
A15:ROOT>ldr special.ndr
ZCPR3 LDR, Version 1.0
 Loading SPECIAL.NDR
```

```
A15:ROOT>pwd
PWD, Version 1.0
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE     A  2: PRIV2       A 15: ROOT

B  0: WORK1       B  1: WORK2       B  2: WORK3       B  4: WORK4
B  5: TEXT        B  6: MAIL

A15:ROOT>priv2:
PW? mypass2

A2:PRIV2>wheel system r
WHEEL, Version 3.0
 Wheel Byte is OFF

A2:PRIV2>pwd pass
PWD, Version 1.0
 Password Request Denied - Not Wheel
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE     A  2: PRIV2       A 15: ROOT

B  0: WORK1       B  1: WORK2       B  2: WORK3       B  4: WORK4
B  5: TEXT        B  6: MAIL
```

3. Command Lines

The following terminal session extracts should be clear about command lines under ZCPR3. Comments are included in the terminal sessions.

```
A0:BASE>; Any Line beginning with a semicolon is a comment
A0:BASE>note Any line whose verb is the word "NOTE" is a comment
A0:BASE>note NOTE is handy to insert comments into lines with more than
A0:BASE>note   one command in them
A0:BASE>note Such lines separate commands with a semicolon


A0:BASE>dir;note I just did a directory display
 RHEX    .COM    2
           A0:BASE --     1 Files Using     2K (  204K Left)


A0:BASE>dir;NOTE This line contains 3 commands (incl one NOTE);dir root:
 RHEX    .COM    2
           A0:BASE --     1 Files Using     2K (  204K Left)
 MYTERM  .Z3T    2r| SPECIAL .NDR    2 | SYS     .ENV    2r| SYS     .FCP    2r
 SYS     .NDR    2r| SYS     .RCP    2r| SYS1    .FCP    2r| SYS1    .RCP    2r
 SYS2    .FCP    2r| SYS2    .RCP    2r| SYS3    .RCP    2r| Z3TCAP  .TCP    8r
            A15:ROOT --    12 Files Using    30K (  204K Left)


A0:BASE>era *.com i;dir;NOTE See the extended options on the basic commands?
 RHEX    .COM - Erase (Y/N)? n
 RHEX    .COM    2
           A0:BASE --     1 Files Using     2K (  204K Left)
```

4. Command Processing

When a ZCPR3 user issues a command, a sequence of events takes place in order to identify that command and execute it.

This sequence is outlined briefly:

1) the command is parsed; the first word in the command line (or subline if semicolons are used to place several commands on one line) is taken to be the name of the command

2) ZCPR3 checks to see if this command is a Flow Command (IF/ELSE/FI/XIF), and, if so, ZCPR3 runs the command

3) ZCPR3 then checks to see if the current IF condition is TRUE; IFs may be nested eight levels deep under ZCPR3; if the current IF condition is TRUE, ZCPR3 continues, else it flushes the command and goes on to the next command

4) ZCPR3 then checks to see if the command is built into the ZCPR3 Command Processor itself; if so, ZCPR3 runs the command

5) ZCPR3 then checks to see if the command is built into the current Resident Command Package (RCP); if so, ZCPR3 runs the command

6) ZCPR3 then searches along a series of directories indicated by a command-search path for a COM file with the same name as the command; if found, ZCPR3 loads the COM file and runs it

7) finally, if all of the above fails, ZCPR3 invokes an error handler or an extended command processor to process the command as an error or to try to resolve it further

Some examples:

```
A0:BASE>work2:

B1:WORK2>dir
            B1:WORK2 --     0 Files Using     0K (  302K Left)
```

```
B1:WORK2>NOTE in many ZCPR3 systems, you will find RCPs -
B1:WORK2>NOTE   Resident Command Packages
B1:WORK2>NOTE this system has several, located in the ROOT
B1:WORK2>dir root:*.rcp
 SYS     .RCP    2r| SYS1    .RCP    2r| SYS2    .RCP    2r| SYS3    .RCP    2r
            A15:ROOT --     4 Files Using     8K (  204K Left)

B1:WORK2>NOTE SYS.RCP is the default RCP I use
B1:WORK2>NOTE the H command tells the user what RCP he has loaded and
B1:WORK2>NOTE    what commands are available in it
B1:WORK2>h
SYS 1.0A
  CP    ECHO  ERA   LIST
  NOTE  P     POKE  PROT
  REN   TYPE

B1:WORK2>NOTE there are 10 commands in this RCP
B1:WORK2>cp work2:=base:rhex.com
 Done

B1:WORK2>dir
 RHEX    .COM    2
           B1:WORK2 --     1 Files Using     2K (  300K Left)

B1:WORK2>cp rhex2.com=rhex.com
 Done

B1:WORK2>dir
 RHEX    .COM    2 | RHEX2   .COM    2
           B1:WORK2 --     2 Files Using     4K (  298K Left)

B1:WORK2>era *.com i
 RHEX    .COM - Erase (Y/N)? n
 RHEX2   .COM - Erase (Y/N)? y

B1:WORK2>cp rhex1.com=rhex.com;cp rhex2.com=rhex.com
 Done
 Done

B1:WORK2>prot *.* r
 RHEX    .COM Set to R/O
 RHEX1   .COM Set to R/O
 RHEX2   .COM Set to R/O

B1:WORK2>dir
 RHEX    .COM    2r| RHEX1   .COM    2r| RHEX2   .COM    2r
           B1:WORK2 --     3 Files Using     6K (  296K Left)

B1:WORK2>prot rhex1.com
 RHEX1   .COM Set to R/W
```

```
B1:WORK2>era *.com
 RHEX    .COM is R/O
 RHEX1   .COM
 RHEX2   .COM is R/O

B1:WORK2>dir
 RHEX    .COM    2r| RHEX2   .COM    2r
            B1:WORK2 --     2 Files Using     4K (  298K Left)

B1:WORK2>echo this command simply echos the command line, as in messages
THIS COMMAND SIMPLY ECHOS THE COMMAND LINE, AS IN MESSAGES

B1:WORK2>ed demo.txt

NEW FILE
     : *i
    1:  This is a test
    2:  This is only a test
    3:
     : *e

B1:WORK2>cp demo2.txt=demo.txt
 Done

B1:WORK2>dir *.txt
 DEMO    .TXT    2 | DEMO2   .TXT    2
            B1:WORK2 --     2 Files Using     4K (  294K Left)

B1:WORK2>ren demo1.txt=demo2.txt

B1:WORK2>cp demo2.txt=demo.txt
 Done

B1:WORK2>dir *.txt
 DEMO    .TXT    2 | DEMO1   .TXT    2 | DEMO2   .TXT    2
            B1:WORK2 --     3 Files Using     6K (  292K Left)

B1:WORK2>ren demo1.txt=demo2.txt
 DEMO1   .TXT - Erase (Y/N)? n

B1:WORK2>type demo.txt

This is a test
This is only a test
```

```
B1:WORK2>type *.txt

This is a test
This is only a test

 Typing  DEMO    .TXT -

This is a test
This is only a test

 Typing  DEMO1   .TXT -

This is a test
This is only a test

B1:WORK2>p 8000 801f;NOTE I look at memory
 Peek at 8000
 8000 -  C3 29 00 C3 CE 80 C3 47 81 C3 82 81 C3 67 81 C3 |C).CN.CG.C..Cg.C|
 8010 -  7E 81 C3 E9 80 C3 22 81 C3 10 81 80 F3 00 00 11 |~.Ci.C".C...s...|

B1:WORK2>p 0 f;NOTE anywhere in memory
 Peek at 0000
 0000 -  C3 03 E2 01 11 C3 06 D4 00 FF 00 FF 00 FF 00 FF |C.b..C.T........|

B1:WORK2>poke 8000 1 2 3 "this is a test
 Poke at 8000

B1:WORK2>p 8000 801f
 Peek at 8000
 8000 -  01 02 03 54 48 49 53 20 49 53 20 41 20 54 45 53 |...THIS IS A TES|
 8010 -  54 81 C3 E9 80 C3 22 81 C3 10 81 80 F3 00 00 11 |T.Ci.C".C...s...|

B1:WORK2>NOTE the RCP commands can be changed by loading a new RCP
B1:WORK2>ldr root:sys3.rcp
ZCPR3 LDR, Version 1.0
 Loading SYS3.RCP
B1:WORK2>h
SYS 1.0C
  CP    ECHO  ERA   NOTE
  P     POKE  REN   TYPE
  WHL   WHLQ

B1:WORK2>cp demo3.txt=demo.txt
 No Wheel

B1:WORK2>era *.txt
 No Wheel

B1:WORK2>wheel system s
WHEEL, Version 3.0
 Wheel Byte is ON
```

```
B1:WORK2>cp demo3.txt=demo.txt
 Done
```

               Some examples of Flow Commands, invoked from
         Flow Command Packages (FCPs) follow:

```
B1:WORK2>NOTE now for Flow Command Packages:
B1:WORK2>NOTE under FCPs, we have IF/ELSE/FI (ENDIF)/XIF (Exit All IFs)
B1:WORK2>NOTE   Flow Commands:

B1:WORK2>if exist demo.txt
 IF T

B1:WORK2>type demo.txt

This is a test
This is only a test

B1:WORK2>else
 IF F

B1:WORK2>type demo2.txt

B1:WORK2>fi
 To No IF

B1:WORK2>if ~exist demo.txt
 IF F

B1:WORK2>type demo.txt

B1:WORK2>else
 IF T

B1:WORK2>type demo2.txt

This is a test
This is only a test

B1:WORK2>fi
 To No IF

B1:WORK2>if exist *.txt
 IF T

B1:WORK2>type demo.txt

This is a test
This is only a test
```

```
B1:WORK2>echo we are in a TRUE IF
WE ARE IN A TRUE IF

B1:WORK2>xif
 To No IF

B1:WORK2>NOTE IFs can be nested up to 8 levels deep:
B1:WORK2>if exist demo.txt
 IF T
B1:WORK2>if exist demo2.txt
 IF T
B1:WORK2>if exist demo.txt
 IF T
B1:WORK2>if exist demo3.txt
 IF T
B1:WORK2>else
 IF F
B1:WORK2>fi
 To IF T
B1:WORK2>fi;fi;fi
 To IF T
 To IF T
 To No IF
```

Command files and command file processors are discussed next. ZEX, a memory-based command file processor, is designed to be the principal tool used.

```
B1:WORK2>NOTE ZEX is the command-file processor, memory-based
B1:WORK2>NOTE   Under ZEX, there is a GOTO command which works
B1:WORK2>NOTE   in conjunction with IFs to provide looping capability
B1:WORK2>ed demo.zex

NEW FILE
     : *i
    1:  NOTE Set Register 1 to 0;reg s1 0
    2:  ;=loop
    3:  NOTE Exit all pending IFs;xif
    4:  NOTE Add 1 to Register 1;reg p1
    5:  NOTE Test for end of loop;if ~1 3
    6:  NOTE Branch to LOOP if Register 1 <> 3;goto loop
    7:  NOTE Done with IF if Register 1 = 3;fi
    8:
     : *e
```

```
B1:WORK2>type demo.zex

NOTE Set Register 1 to 0;reg s1 0
;=loop
NOTE Exit all pending IFs;xif
NOTE Add 1 to Register 1;reg p1
NOTE Test for end of loop;if ~1 3
NOTE Branch to LOOP if Register 1 <> 3;goto loop
NOTE Done with IF if Register 1 = 3;fi
```

Here is an actual run of a ZEX command file (DEMO.ZEX) which illustrates looping:

```
B1:WORK2>zex demo
ZEX, Version 3.0

                                 -- Pass 1 --

B1:WORK2> ZEX: NOTE Set Register 1 to 0;reg s1 0
REG, Version 1.0
 Reg 1 =   0
B1:WORK2> ZEX: ;=loop
B1:WORK2> ZEX: NOTE Exit all pending IFs;xif
 To No IF
B1:WORK2> ZEX: NOTE Add 1 to Register 1;reg p1
REG, Version 1.0
 Reg 1 =   1
B1:WORK2> ZEX: NOTE Test for end of loop;if ~1 3
 IF T
B1:WORK2> ZEX: NOTE Branch to LOOP if Register 1 <> 3;goto loop
 GOTO Label LOOP

                                 -- Pass 2 --

B1:WORK2> ZEX: NOTE Exit all pending IFs;xif
 To No IF
B1:WORK2> ZEX: NOTE Add 1 to Register 1;reg p1
REG, Version 1.0
 Reg 1 =   2
B1:WORK2> ZEX: NOTE Test for end of loop;if ~1 3
 IF T
B1:WORK2> ZEX: NOTE Branch to LOOP if Register 1 <> 3;goto loop
 GOTO Label LOOP
```

```
                                -- Pass 3 --

B1:WORK2> ZEX: NOTE Exit all pending IFs;xif
 To No IF
B1:WORK2> ZEX: NOTE Add 1 to Register 1;reg p1
REG, Version 1.0
 Reg 1 =   3
B1:WORK2> ZEX: NOTE Test for end of loop;if ~1 3
 IF F

                                 -- Done --

B1:WORK2> ZEX: NOTE Branch to LOOP if Register 1 <> 4;goto loop
B1:WORK2> ZEX: NOTE Done with IF if Register 1 = 3;fi
 To No IF
B1:WORK2> ZEX: Done>
```

The example above was for academic purposes. Two examples of ZEX command files which I use every day follow. One command file assembles programs for me using the MAC assembler, and the other uses the M80 assembler with none, one, two, three, or four libraries, generating different command lines depending upon how many libraries were specified in the original command line.

---- Command File for MAC Assembly ----

```
;  MAC -- CP/M Standard MACRO Assembler and Loader
MAC $1 $$PZ SZ
IF INPUT Type N or F to Abort if Errors Exist
ERA $1.BAK
ERA $1.COM
MLOAD $1
FI
ERA $1.HEX
;  Assembly Complete
```

---- Command File for M80 Assembly ----

```
;
;  M80.ZEX -- MACRO-80 Assembler and Linker
;     Up to 4 Libraries Specified
;
;  ^& Suppress FALSE IF Printout
;
if nul $1 ;note Print Error Message
echo      **** No Parameter Specified ****
else      ;note Perform Assembly
M80 =$1
if input Type T to Continue or F to Abort (in case of Errors)
ERA $1.BAK
ERA $1.COM
if ~nul $5 ;note Link 4 Additional Libraries
L80 /P:100,$1,$2/S,$3/S,$4/S,$5/S,A:Z3LIB/S,A:SYSLIB/S,$1/N,/U,/E
goto done
fi
if ~nul $4 ;note Link 3 Additional Libraries
L80 /P:100,$1,$2/S,$3/S,$4/S,A:Z3LIB/S,A:SYSLIB/S,$1/N,/U,/E
goto done
fi
if ~nul $3 ;note Link 2 Additional Libraries
L80 /P:100,$1,$2/S,$3/S,A:Z3LIB/S,A:SYSLIB/S,$1/N,/U,/E
goto done
fi
if ~nul $2 ;note Link 1 Additional Library
L80 /P:100,$1,$2/S,A:Z3LIB/S,A:SYSLIB/S,$1/N,/U,/E
goto done
else       ;note Standard Link
L80 /P:100,$1,A:Z3LIB/S,A:SYSLIB/S,$1/N,/U,/E
;=done          Done with Link
fi         ;note on IF ~NUL Tests
fi         ;note on IF INPUT
ERA $1.REL
fi         ;note on IF NUL
;
;  Assembly Complete
;
```

5. Error Handlers

Error Handlers are programs which handle command line errors in a "nice" way. They may be used anywhere, including within ZEX command files. A few examples:

```
B1:WORK2>NOTE There are a number of error handlers on this system:
B1:WORK2>dir root:error?.com s
 ERROR1  .COM    2r| ERROR2  .COM    4r| ERROR3  .COM    2r| ERROR4  .COM    2r
             A15:ROOT --     4 Files Using    10K (  204K Left)
```

Error Handlers are installed by simply giving their name.

```
B1:WORK2>error4
ERROR4, Version 1.0
 Error Handler Installed

B1:WORK2>NOTE ERROR4 is a simpler error handler
B1:WORK2>NOTE with the invalid command "XXXX";xxxx

 File XXXX.COM Not Found

B1:WORK2>NOTE ERROR4 simply says what happened
```

Error Handlers may vary in features and complexity. ERROR1 is one of the more complex. ERROR2, by the way, is a screen-oriented version of ERROR1, using reverse video and cursor addressing. See the section on Z3TCAP later for more details.

```
B1:WORK2>error1;NOTE ERROR1 is a more sophisticated error handler
ERROR1, Version 1.0
 Error Handler Installed
```

```
B1:WORK2>xxxx

ERROR1, Version 1.0

Error Line is:
   XXXX

Options are:
 1. Replace Command in Error with a New Command
     Replace XXXX
 2. Advance to Next Command and Resume Processing
     Advance to
 3. Replace Entire Line with a New Line
     Replace XXXX
 4. Throw Away Entire Line and Continue
     Throw Away XXXX

Select Option - 1
Replacement Command?
  dir
 DEMO    .BAK    0 | DEMO    .TXT    2 | DEMO    .ZEX    2 | DEMO1   .TXT    2
 DEMO2   .TXT    2 | DEMO3   .TXT    2 | RHEX    .COM    2r| RHEX2   .COM    2r
             B1:WORK2 --     8 Files Using    14K (  288K Left)
B1:WORK2>xxxx;dir *.com

ERROR1, Version 1.0

Error Line is:
   XXXX;DIR *.COM

Options are:
 1. Replace Command in Error with a New Command
     Replace XXXX
 2. Advance to Next Command and Resume Processing
     Advance to DIR *.COM
 3. Replace Entire Line with a New Line
     Replace XXXX;DIR *.COM
 4. Throw Away Entire Line and Continue
     Throw Away XXXX;DIR *.COM

Select Option - 2

 RHEX    .COM    2r| RHEX2   .COM    2r
            B1:WORK2 --     2 Files Using     4K (  288K Left)
```

6. Aliases

Aliases are COM files created by the ALIAS command which contain one or more command lines which are invoked when the Alias name is typed. Parameter passing into the command lines within an Alias is supported in a manner similar to command file parameter passing. Aliases are convenient to create command scripts which are used repeatedly, and the special commands, such as STARTUP (used on cold boot to run a series of programs to initialize the system), are created as Aliases.

```
B1:WORK2>NOTE you have to be a WHEEL to create ALIASes
B1:WORK2>wheel /s
WHEEL, Version 3.0
 Wheel Password?  Wheel Byte is ON

B1:WORK2>NOTE a number of parameters and some information can be determined
B1:WORK2>NOTE   and expanded by an alias
B1:WORK2>alias
ALIAS, Version 1.0

 Input Alias (RETURN to Abort)
 --> echo The name of this Alias is $0;      <-- I ended these
echo The current DU is $d$u:;                <-- lines with ^E
echo and the first 4 parameters are:;
echo $1 $2 $3 $4
 Name of ALIAS Command (RETURN to Abort)? cmdstat
 Alias Created

B1:WORK2>NOTE the alias is a very short file (under 2K)
B1:WORK2>dir cmdstat.com
 CMDSTAT .COM    2
            B1:WORK2 --     1 Files Using     2K (  292K Left)

B1:WORK2>cmdstat

THE NAME OF THIS ALIAS IS CMDSTAT
THE CURRENT DU IS B1:
AND THE FIRST 4 PARAMETERS ARE:

B1:WORK2>cmdstat this is a very short demo

THE NAME OF THIS ALIAS IS CMDSTAT
THE CURRENT DU IS B1:
AND THE FIRST 4 PARAMETERS ARE:
THIS IS A VERY
```

```
B1:WORK2>cmdstat hello, world

THE NAME OF THIS ALIAS IS CMDSTAT
THE CURRENT DU IS B1:
AND THE FIRST 4 PARAMETERS ARE:
HELLO, WORLD

B1:WORK2>NOTE aliases are convenient for a number of things --
B1:WORK2>NOTE   they are intended primarily to replace tedious command
B1:WORK2>NOTE   sequences with a simple command
B1:WORK2>alias
ALIAS, Version 1.0

 Input Alias (RETURN to Abort)
 --> dir $1;era $1 i;dir $1
 Name of ALIAS Command (RETURN to Abort)? exera
 Alias Created

B1:WORK2>NOTE I now have an ALIAS which displays a directory of selected
B1:WORK2>NOTE   files, allows me to erase them with inspection, and then
B1:WORK2>NOTE   displays the same directory again to let me see the
B1:WORK2>NOTE   results
B1:WORK2>dir
 CMDSTAT .COM    2 | DEMO    .TXT    2 | DEMO    .ZEX    2 | DEMO1   .TXT    2
 DEMO2   .TXT    2 | DEMO3   .TXT    2 | EXERA   .COM    2 | RHEX    .COM    2r
 RHEX2   .COM    2r
           B1:WORK2 --     9 Files Using    18K (  284K Left)
```

The following runs an Alias:

```
B1:WORK2>exera demo?.txt

 DEMO    .TXT    2 | DEMO1   .TXT    2 | DEMO2   .TXT    2 | DEMO3   .TXT    2
            B1:WORK2 --     4 Files Using     8K (  284K Left)
 DEMO    .TXT - Erase (Y/N)? n
 DEMO1   .TXT - Erase (Y/N)? y
 DEMO2   .TXT - Erase (Y/N)? y
 DEMO3   .TXT - Erase (Y/N)? n
 DEMO    .TXT    2 | DEMO3   .TXT    2
           B1:WORK2 --     2 Files Using     4K (  288K Left)

B1:WORK2>exera demo3.txt

 DEMO3   .TXT    2
           B1:WORK2 --     1 Files Using     2K (  288K Left)
 DEMO3   .TXT - Erase (Y/N)? y
           B1:WORK2 --     0 Files Using     0K (  290K Left)
```

```
B1:WORK2>NOTE also, since IFs are everywhere, they can be used in aliases:
B1:WORK2>alias
ALIAS, Version 1.0

 Input Alias (RETURN to Abort)
 --> if exist $1;type $1 p;fi
 Name of ALIAS Command (RETURN to Abort)? typeit
 Alias Created

B1:WORK2>typeit demo.txt

 IF T

This is a test
This is only a test

 To No IF

B1:WORK2>cp demo1.txt=demo.txt
 Done

B1:WORK2>dir demo?.txt
 DEMO    .TXT    2 | DEMO1   .TXT    2
            B1:WORK2 --     2 Files Using     4K (  286K Left)

B1:WORK2>typeit demo?.txt

 IF T

This is a test
This is only a test

 Typing  DEMO    .TXT -

This is a test
This is only a test

 To No IF

B1:WORK2>typeit nofile.txt

 IF F
 To No IF
```

```
B1:WORK2>NOTE or I can expand TYPEIT to be better
B1:WORK2>alias typeit
ALIAS, Version 1.0
 Alias Name: TYPEIT
 Old Alias Command Line:
  1 --> IF EXIST $1;
  2 --> TYPE $1 P;
  3 --> FI

 Input Alias (RETURN to Abort)
 --> if exist $1;type $1 p;else;echo file $1 not found;fi
 File TYPEIT  .COM Exists - Overwrite (Y/N)? Y

 Alias Created

B1:WORK2>typeit demo.txt

 IF T

This is a test
This is only a test

 IF F
 To No IF

B1:WORK2>typeit nofile.txt

 IF F
 IF T
FILE NOFILE.TXT NOT FOUND
 To No IF
```

7. S h e l l s

ZCPR3 Shells are front-ends which provide a user interface in place of the normal ZCPR3 prompt. The following terminal sessions show the MENU and SH shells in action.

```
B1:WORK2>NOTE Shells are Front-End Processors which can run in place
B1:WORK2>NOTE   of the ZCPR3 Command Processor
B1:WORK2>NOTE Actually, the ZCPR3 Command Processor is still being
B1:WORK2>NOTE   used, but it is transparent to the user now
B1:WORK2>NOTE Two shells I am going to demonstrate now are MENU and
B1:WORK2>NOTE   SH:

B1:WORK2>dir root:menu.* a;dir root:sh*.* a
 MENU    .COM    4r
           A15:ROOT --     1 Files Using     4K (  204K Left)
 SH      .COM    4r| SHDEFINE.COM    4r| SHFILE  .COM    2r| SHOW    .COM    4r
 SHVAR   .COM    4r
           A15:ROOT --     5 Files Using    18K (  204K Left)

B1:WORK2>NOTE The MENU shell consists of only MENU.COM
B1:WORK2>NOTE The SH shell is SH.COM, but can use SHDEFINE, SHFILE, and SHVAR
B1:WORK2>NOTE   for support
B1:WORK2>NOTE First, MENU:

B1:WORK2>ed menu.cpr

NEW FILE
     : *i
    1:  -dx
    2:  #
    3:       Sample Menu
    4:   D - Directory Display
    5:   Z - Run Any ZCPR3 Command
    6:
    7:   1 - Set Name of Working File (Currently $f1)
    8:   2 - Edit Working File
    9:   3 - Type Working File
   10:  #
   11:  d!dir
   12:  z!"Enter Command Line -- "
   13:  1setfile 1 "Enter File Name -- "
   14:  2ed $f1
   15:  3!type $f1
   16:  ##
   17:
     : *e
```

To run the MENU shell, just give its name.

```
B1:WORK2>menu
 Shell Installed
MENU  Version 3.0
     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently .)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - D
 CMDSTAT .COM    2 | MENU    .BAK    0 | DEMO    .TXT    2 | DEMO    .ZEX    2
 DEMO1   .TXT    2 | EXERA   .COM    2 | MENU    .CPR    2 | RHEX    .COM    2r
 RHEX2   .COM    2r| TYPEIT  .COM    2
            B1:WORK2 --    10 Files Using    18K (  284K Left)


MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently .)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - Z
Enter Command Line -- dir *.com;era *.com i
 CMDSTAT .COM    2 | EXERA   .COM    2 | RHEX    .COM    2r| RHEX2   .COM    2r
 TYPEIT  .COM    2
            B1:WORK2 --     5 Files Using    10K (  284K Left)
 CMDSTAT .COM - Erase (Y/N)? y
 EXERA   .COM - Erase (Y/N)? y
 RHEX    .COM is R/O
 RHEX2   .COM is R/O
 TYPEIT  .COM - Erase (Y/N)? y

MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently .)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - Z
Enter Command Line -- prot rhex?.com;era rhex?.com
 RHEX    .COM Set to R/W
 RHEX2   .COM Set to R/W
 RHEX    .COM
 RHEX2   .COM

MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently .)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - D
 DEMO    .BAK    0 | DEMO    .TXT    2 | DEMO    .ZEX    2 | DEMO1   .TXT    2
 MENU    .CPR    2
           B1:WORK2 --     5 Files Using     8K (  294K Left)
```

MENU supports up to 4 file names which can be used as variables within MENU. The common application is to use these files names to specify working files.

```
MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently .)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 1
Enter File Name -- myfile.txt
SETFILE, Version 1.0
 File Name 1 is MYFILE  .TXT


MENU  Version 3.0
     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 2

NEW FILE
     : *i
    1:  This is MYFILE.TXT
    2:  Isn't this fun?
    3:
     : *b0p
    1:  This is MYFILE.TXT
    2:  Isn't this fun?
    1: *e

MENU  Version 3.0
     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 3

This is MYFILE.TXT
Isn’t this fun?

MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 2
     : *#a
    1: *i
    1:  I have modified MYFILE.TXT
    2:
    2: *b0p
    1:  I have modified MYFILE.TXT
    2:  This is MYFILE.TXT
    3:  Isn’t this fun?
    1: *e

MENU  Version 3.0
     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 3

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?
```

```
MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - ^C
B1:WORK2>
```

The following is a demonstration of the ZCPR3 Named-Variable Shell, SH. SH allows, among other things, the user to specify variables in his command lines which are expanded as macros when the command lines are interpreted by SH. SH then passes the expanded command lines to the ZCPR3 Command Processor, which executes them.

```
B1:WORK2>NOTE Now I will demonstrate SH
B1:WORK2>sh
Shell Installed
B1:WORK2>> ;first, SH looks like the normal ZCPR3, except that the
B1:WORK2>> ;prompt is >>
B1:WORK2>>
B1:WORK2>> ;commands run normally under SH:

B1:WORK2>> dir *.txt
 DEMO    .TXT    2 | DEMO1   .TXT    2 | MYFILE  .TXT    2
            B1:WORK2 --     3 Files Using     6K (  292K Left)

B1:WORK2>> error4
ERROR4, Version 1.0
 Error Handler Installed

B1:WORK2>> NOTE Shells, like many things under ZCPR3, can be nested:
B1:WORK2>> menu
 Shell Installed
MENU  Version 3.0
     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 3

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?

MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - Z
Enter Command Line -- NOTE and, when I exit, I’m back to SH
```

```
MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - ^C


B1:WORK2>> ; SH has some built-in commands, which can be determined by
B1:WORK2>> ; a ? command:
B1:WORK2>> ?
SH Commands --
  ?         SHCMT     SHECHO    SHEXIT


B1:WORK2>> ; guess what SHEXIT does:
B1:WORK2>> shexit
Exiting Shell

B1:WORK2>NOTE oh, well, back to ZCPR3 ... but we were talking
B1:WORK2>NOTE about SH:
B1:WORK2>sh
Shell Installed

B1:WORK2>> ; SHCMT is intended to switch SH into a comment
B1:WORK2>> ; mode, for times like this when I want to record
B1:WORK2>> ; a lot of text and a few commands:
B1:WORK2>> shcmt

B1:WORK2; note that the prompt is now "B1:WORK2; "
B1:WORK2; I don’t have to type the leading ; or the word NOTE
B1:WORK2;
B1:WORK2; If I want to execute a command, I simply prefix it with
B1:WORK2; an exclamation mark:
B1:WORK2; !dir *.txt
 DEMO    .TXT    2 | DEMO1   .TXT    2 | MYFILE  .TXT    2
            B1:WORK2 --     3 Files Using     6K (  292K Left)

B1:WORK2; !menu
 Shell Installed
MENU  Version 3.0
     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - 3

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?
```

```
MENU  Version 3.0 Strike Any Key -

     Sample Menu
 D - Directory Display
 Z - Run Any ZCPR3 Command

 1 - Set Name of Working File (Currently MYFILE.TXT)
 2 - Edit Working File
 3 - Type Working File
Command (<CR>=Menu, ^C=ZCPR3) - ^C

B1:WORK2; and we are back:
B1:WORK2; !?
SH Commands --
  ?         SHCMT     SHECHO    SHEXIT

B1:WORK2; all commands work that way under SH
B1:WORK2; as I mentioned, SH is a Variable Shell
B1:WORK2; by this I mean that it supports named variables, which
B1:WORK2; can be defined (in groups) by SHDEFINE or one at a time
B1:WORK2; by SHVAR
B1:WORK2;
B1:WORK2; SHVAR with no args displays the names of the current
B1:WORK2; variables
B1:WORK2; !shvar
SHVAR, Version 1.0
 Shell Variables --
  -- No Variables Defined --


B1:WORK2;
B1:WORK2; with an arg (actually, 2 args), SHVAR defines variables
B1:WORK2; !shvar file1 myfile.txt
SHVAR, Version 1.0
 Shell Variable FILE1 = MYFILE.TXT
 Writing Shell Variable File SH      .VAR


B1:WORK2;
B1:WORK2; and now I can reference variables by preceeding them with
B1:WORK2; a % character
B1:WORK2; !type %file1

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?
```

```
B1:WORK2; does the same as
B1:WORK2; !type myfile.txt

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?

B1:WORK2; note that SH variables can only be used under SH
B1:WORK2; don’t confuse these with aliases, which can be used
B1:WORK2; anywhere, including under SH
B1:WORK2; !alias
ALIAS, Version 1.0

 Input Alias (RETURN to Abort)
 --> echo hello, world - my name is $0
 Name of ALIAS Command (RETURN to Abort)? hello
 Alias Created

B1:WORK2; !hello

HELLO, WORLD - MY NAME IS HELLO

B1:WORK2; !shexit
Exiting Shell

B1:WORK2>hello

HELLO, WORLD - MY NAME IS HELLO

B1:WORK2>sh
Shell Installed

B1:WORK2>> shcmt
B1:WORK2; also, SH variables can be referenced by other SH variables,
B1:WORK2; up to 20 levels deep:
B1:WORK2; !shvar cmddemo type %%file1
SHVAR, Version 1.0
 Shell Variable CMDDEMO = TYPE %FILE1
 Writing Shell Variable File SH      .VAR

B1:WORK2; note my use of the double %% to indicate that I wanted
B1:WORK2; the % character substituted -- If I used just 1 %, then
B1:WORK2; the value of the variable would be substituted:


B1:WORK2; !shvar cmddemo1 type %file1
SHVAR, Version 1.0
 Shell Variable CMDDEMO1 = TYPE MYFILE.TXT
 Writing Shell Variable File SH      .VAR
```

```
B1:WORK2; see the difference?
B1:WORK2; so, to execute:
B1:WORK2; !%cmddemo

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?

B1:WORK2; !%cmddemo1

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?

B1:WORK2; as a side comment, the SHECHO command can be used to make
B1:WORK2; SH show you the command line it is generating:
B1:WORK2; !shecho
 Echo of Shell Commands is ON

B1:WORK2; !%cmddemo
TYPE MYFILE.TXT

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?

B1:WORK2; Now, if I change the definition of FILE1:
B1:WORK2; !shvar file1 hisfile.txt
SHVAR FILE1 HISFILE.TXT
SHVAR, Version 1.0
 Shell Variable FILE1 = HISFILE.TXT
 Writing Shell Variable File SH      .VAR

B1:WORK2; the meaning of CMDDEMO is different:
B1:WORK2; !%cmddemo
TYPE HISFILE.TXT
 No Files

B1:WORK2; while CMDDEMO1 remains unchanged
B1:WORK2; !%cmddemo1
TYPE MYFILE.TXT

I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?
```

```
B1:WORK2; !ed %file1
ED HISFILE.TXT

NEW FILE
     : *i
    1:  This is HISFILE.TXT
    2:
     : *e

B1:WORK2; !%cmddemo;%cmddemo1
TYPE HISFILE.TXT;TYPE MYFILE.TXT

This is HISFILE.TXT


I have modified MYFILE.TXT
This is MYFILE.TXT
Isn’t this fun?

B1:WORK2; and so on ...

B1:WORK2; !shexit
Exiting Shell
B1:WORK2>
```

8. Z 3 T C A P

The ZCPR3 TCAP (Terminal Capability) Facility (Z3TCAP) allows ZCPR3 to have a variety of easily-transportable screen-oriented utilities. ERROR2, VFILER, SHOW, and VMENU are such utilities found under the ZCPR3 System.

```
B1:WORK2>NOTE The ZCPR3 TCAP (Z3TCAP) facility is supported by
B1:WORK2>NOTE   three programs and one data file:
B1:WORK2>dir root:tc*.com a;dir root:*.tcp
 TCCHECK .COM    2r| TCMAKE  .COM    6r| TCSELECT.COM    4r
            A15:ROOT --     3 Files Using    12K (  202K Left)
 Z3TCAP  .TCP    8r
            A15:ROOT --     1 Files Using     8K (  202K Left)


B1:WORK2>NOTE TCCHECK is used to check the validity of Z3TCAP.TCP:
B1:WORK2>root:
A15:ROOT>tccheck
TCCHECK, Version 1.0
Z3TCAP File Check of Z3TCAP  .TCP Version 1.0
        File Checks with    43 Terminals Defined
```

Over forty terminals are currently supported under the Z3TCAP. Their selection and installation into a ZCPR3 System is illustrated:

```
A15:ROOT>work2:
B1:WORK2>NOTE TCSELECT is used to select your terminal from one of the
B1:WORK2>NOTE  terminals in Z3TCAP.TCP:
```

```
B1:WORK2>tcselect myterm
TCSELECT, Version 1.0

** Terminal Menu 1 for Z3TCAP Version 1.0  **

A.  AA Ambassador            K.  Concept 100
B.  ADDS Consul 980          L.  Concept 108
C.  ADDS Regent 20           M.  CT82
D.  ADDS Viewpoint           N.  DEC VT52
E.  ADM 2                    O.  DEC VT100
F.  ADM 31                   P.  Dialogue 80
G.  ADM 3A                   Q.  Direct 800/A
H.  ADM 42                   R.  General Trm 100A
I.  Bantam 550               S.  Hazeltine 1420
J.  CDC 456                  T.  Hazeltine 1500

Enter Selection, + for Next, or ^C to Exit - +

** Terminal Menu 2 for Z3TCAP Version 1.0  **

A.  Hazeltine 1510           K.  SOROC 120
B.  Hazeltine 1520           L.  Super Bee
C.  Heathkit H19             M.  TAB 132
D.  HP 2621                  N.  Teleray 1061
E.  IBM 3101                 O.  Teleray 3800
F.  Micro Bee                P.  TTY 4424
G.  Microterm ACT IV         Q.  TVI 912
H.  Microterm ACT V          R.  TVI 920
I.  P Elmer 1100             S.  TVI 950
J.  P Elmer 1200             T.  VC 404

Enter Selection, - for Last, + for Next, or ^C to Exit - +

** Terminal Menu 3 for Z3TCAP Version 1.0  **

A.  VC 415
B.  Visual 200
C.  WYSE 50
```

```
Enter Selection, - for Last, or ^C to Exit - -

** Terminal Menu 2 for Z3TCAP Version 1.0  **

A.  Hazeltine 1510          K.  SOROC 120
B.  Hazeltine 1520          L.  Super Bee
C.  Heathkit H19            M.  TAB 132
D.  HP 2621                 N.  Teleray 1061
E.  IBM 3101                O.  Teleray 3800
F.  Micro Bee               P.  TTY 4424
G.  Microterm ACT IV        Q.  TVI 912
H.  Microterm ACT V         R.  TVI 920
I.  P Elmer 1100            S.  TVI 950
J.  P Elmer 1200            T.  VC 404

Enter Selection, - for Last, + for Next, or ^C to Exit - S

  Selected Terminal is: TVI 950                  -- Confirm (Y/N)? Y

File MYTERM  .Z3T Created
B1:WORK2>dir *.z3t
 MYTERM  .Z3T    2
           B1:WORK2 --     1 Files Using     2K (  286K Left)


B1:WORK2>NOTE Once you have a Z3T file, LDR can load it and, at this
B1:WORK2>NOTE   time (after loading), your terminal will be known
B1:WORK2>NOTE   to the ZCPR3 system and the ZCPR3 utilities can
B1:WORK2>NOTE   make use of its features, such as cursor positioning,
B1:WORK2>NOTE   reverse video, arrow keys, etc
B1:WORK2>ldr myterm.z3t
ZCPR3 LDR, Version 1.0
 Loading MYTERM.Z3T

B1:WORK2>NOTE The commands SHOW and VFILER are now configured for
B1:WORK2>NOTE   a TVI 950, as per my selection
```

Not everyone's terminal will be in the default Z3TCAP. To meet this problem, the utility TCMAKE is available.

```
B1:WORK2>NOTE If you terminal is not on the list of terminals in
B1:WORK2>NOTE   Z3TCAP.TCP, then TCMAKE can be used to define it
B1:WORK2>NOTE   I will define my TVI 950 here:
```

```
B1:WORK2>tcmake myterm1
TCMAKE, Version 1.0


        ** Z3TCAP Main Menu for File MYTERM1 .Z3T **

Define: 1. Clear Screen Sequence
        2. Cursor Motion Sequence
        3. Clear to End of Line Sequence
        4. Standout Mode Sequences
        5. Terminal Init/Deinit Sequences
        6. Arrow Keys
        7. Terminal Name

Status: S. Print Status (Definitions so far)

Exit:   X. Exit and Write File
        Q. Quit and Abort Program without Writing File

Command? 1
```

I won't bore you with details here. The terminal session is quite long, illustrating the major features of TCMAKE.

9. "Secure" Systems

With password protecton and named directories (DIR form) built into ZCPR3, ZCPR3 offers a much more secure environment than CP/M. In particular, if the DU form is disabled, the only directories a user can access are those he can name, and some of those may have password protections on them.

Here is a complete session:

```
AMPRO 51K TPA CP/M 2.2 with ZCPR 3.0
 BIOS Version 1.2 on March 24, 1984

ZCPR3 LDR, Version 1.0
 Loading SYS.ENV
 Loading SYS.NDR
 Loading SYS.FCP
 Loading SYS.RCP
ERROR4, Version 1.0
 Error Handler Installed
 WELCOME TO ZCPR III

BASE>dir
 AMPZ358R.COM   10 | SYS3R   .RCP    2
           A0:BASE --     2 Files Using    12K (  266K Left)

BASE>pwd
PWD, Version 1.0
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE1    A  2: PRIVATE2    A 15: ROOT

B  0: DEMO1       B  1: DEMO2       B  2: DEMO3       B  3: DEMO4
B  4: DEMO5       B  5: INTRO       B  6: MAIL

BASE>dir root:
PW? unknown
 AMPZ358R.COM   10 | SYS3R   .RCP    2
           A0:BASE --     2 Files Using    12K (  266K Left)

BASE>dir root:
PW? rpass
 DIR     .COM    2 | ERROR4  .COM    2 | GOTO    .COM    2 | LDR     .COM    4
 MENU    .COM    4 | MKDIR   .COM    6 | PWD     .COM    2 | SETFILE .COM    2
 SH      .COM    4 | SHDEFINE.COM    4 | SHFILE  .COM    2 | SHOW    .COM    4
 SHVAR   .COM    4 | SPECIAL .NDR    2 | STARTUP .COM    2 | SYS     .ENV    2
 SYS     .FCP    2 | SYS     .NDR    2 | SYS     .RCP    2 | TCCHECK .COM    2
 TCMAKE  .COM    6 | TCSELECT.COM    4 | WHEEL   .COM    2 | Z3TCAP  .TCP    8
 ZEX     .COM    6
           A15:ROOT --    25 Files Using    82K (  266K Left)
BASE>xxx

 File XXX.COM Not Found
```

Note that the DU form is simply ignored. No change to files or directory location is made.

```
BASE>1:
BASE>a:
BASE>b:
BASE>dir 1:
 AMPZ358R.COM   10 | SYS3R   .RCP    2
            A0:BASE --     2 Files Using    12K (  266K Left)

BASE>dir demo1:
 AMPZ3-58.COM   10 | AMPZ3-60.COM   10 | AMPZ3-61.COM   10 | AMPZ358R.COM   10
 BDOS58  .COM    4 | BDOS60  .COM    4 | BDOS61  .COM    4 | CPM58   .COM   10
 CPM60   .COM   10 | CPM61   .COM   10 | SYS3R   .RCP    2
            B0:DEMO1 --    11 Files Using    84K (  284K Left)

BASE>demo1:
DEMO1>root:
PW? rpass

ROOT>wheel /s
WHEEL, Version 3.0
 Wheel Password?  Wheel Byte is ON

ROOT>NOTE We now have one directory structure:
ROOT>pwd
PWD, Version 1.0
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE1    A  2: PRIVATE2    A 15: ROOT

B  0: DEMO1       B  1: DEMO2       B  2: DEMO3       B  3: DEMO4
B  4: DEMO5       B  5: INTRO       B  6: MAIL
```

With the ability to have several named directory files, we can have several sets of directories, including some directories which are both hidden and totally inaccessable to the user unless he has the ability to load the proper named directory (NDR) file.

```
ROOT>NOTE Now that I am a WHEEL and in ROOT, I can define another
ROOT>NOTE   directory structure which is special:
```

```
ROOT>ldr special.ndr
ZCPR3 LDR, Version 1.0
 Loading SPECIAL.NDR
ROOT>pwd
PWD, Version 1.0
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE1    A  2: PRIVATE2    A 14: SYSROOT
A 15: ROOT

B  0: DEMO1       B  1: DEMO2       B  2: DEMO3       B  3: DEMO4
B  4: DEMO5       B  5: INTRO       B  6: MAIL

ROOT>NOTE Note that there is a 2nd root, called SYSROOT, which
ROOT>NOTE   was not known (OR ACCESSIBLE) under the old system
ROOT>NOTE   (SYS.NDR)
ROOT>

ROOT>NOTE Also, as a wheel, I can obtain passwords:

ROOT>pwd pass
PWD, Version 1.0
 DU : DIR Name - Password     DU : DIR Name - Password
----  --------   --------    ----  --------   --------
A  0: BASE     -             A  1: PRIVATE1 - MYPASS1
A  2: PRIVATE2 - PASS        A 14: SYSROOT  - SPASS
A 15: ROOT     - RPASS

B  0: DEMO1    -             B  1: DEMO2    -
B  2: DEMO3    -             B  3: DEMO4    -
B  4: DEMO5    -             B  5: INTRO    -
B  6: MAIL     - MPASS

ROOT>private1:
PW? mypass1
PRIVATE1>wheel /r
WHEEL, Version 3.0
 Wheel Password?  Wheel Byte is OFF

PRIVATE1>pwd pass
PWD, Version 1.0
 Password Request Denied - Not Wheel
 DU : DIR Name     DU : DIR Name     DU : DIR Name     DU : DIR Name
----  --------    ----  --------    ----  --------    ----  --------
A  0: BASE        A  1: PRIVATE1    A  2: PRIVATE2    A 14: SYSROOT
A 15: ROOT

B  0: DEMO1       B  1: DEMO2       B  2: DEMO3       B  3: DEMO4
B  4: DEMO5       B  5: INTRO       B  6: MAIL
```

```
PRIVATE1>ldr root:sys.ndr
PW? rpass
ZCPR3 LDR, Version 1.0
 Loading SYS.NDR

PRIVATE1>ldr root:special.ndr
PW? rpass
ZCPR3 LDR, Version 1.0
 Loading SPECIAL.NDR

PRIVATE1>sysroot:
PW? spass

SYSROOT>root:
PW? rpass

ROOT>ldr sys.ndr
ZCPR3 LDR, Version 1.0
 Loading SYS.NDR

ROOT>sysroot:
ROOT>NOTE SYSROOT is not even defined now
```

---- End of Introduction to ZCPR3 ----

T A B L E O F C O N T E N T S