4404 Operating System

Version 1.5 Notes

ABOUT THIS DOCUMENT

This document consists of two sections that detail the manual changes and errata in the 4404 Reference and User's manuals. Some changes were caused by the introduction of V1.5, while others were inaccuracies in the original documentation. You should either write the changes into your copy of the two manuals, or mark the affected pages with a note to see this document.

4404 REFERENCE MANUAL CHANGES AND ERRATA

INTRODUCTION

Version 1.5 of the 4404 Operating System adds functionality to the operating system. This document covers these additions and corrects some inaccuracies in the 4404 Reference Manual. Please update your Reference Manual to reflect these changes and corrections.

4404 OPERATING SYSTEM ENVIRONMENT NOTES

Version 1.5 of the 4404 Operating System now stores environment information pointers after the program argument pointers. These pointers are stored as follows:

- program argument count — by convention, *argc* in most 'C' programs
- pointer to first argument (program name) — by convention, *argv[0]*
- pointer to second argument — *argv[1]*
- ... (following argument pointers)
- ponter to the *(argc)th* argument — *argv[argc]*
- null pointer — *argv[argc+1]*
- pointer to first environment string — *argv[(argc+1)+1]*
- pointer to second environment string — *argv[(argc+1)+2]*
- ... (following environment string pointers)
- pointer to the *n*th environment string — *argv[(argc+1)+n]*
- null pointer — *argv[(argc+1)+(n+1)]*
- argument strings — These are the strings pointed to by argv[0] through argv[argc]. Each string is terminated with a null.
- extra zero byte — an additional null past the terminal null of the final argument string.
- environment strings — the null-terminated strings pointed to by the environment string pointers are stored past this point.
- extra zero byte — an additional null after the terminal null of the last environment string terminates the data structure

If there is no environment, then there are two zero longwords after the argument pointers. (This implies that even if there is no environment, the C variable "environ" is nonzero, pointing to the zero longword).

Environments are initially created by the *login* program. The default environment contains the following named variables:

HOME — HOME contains the name of the logged-in user's "home" directory — the one in which the user is located when first logged-in. That is, for user *foo*, HOME contains */foo*.

PATH — PATH is the search path *shell* uses to locate files (usually commands). This is normally the current working directory, then the directory */bin*. User *system* also includes the directory */etc* in the PATH.

TERM — TERM is the terminal type (the default is *4404* as specified by the login invocation of /bin/shell).

The *addpath* and *rmpath* commands also affect the PATH environment string. Adding another search *path* to the PATH environment string is done by using the *addpath* shell command. Removing a search *path* from the PATH environment string is done by using the *rmpath* shell command. The command in the 4404 *shell* for manipulating environments is the *shell* command *set*.

The command in *script* for manipulating environments is the command *env*. Invoking *env* with no arguments lists all of the environment strings known to *script*. Specifying an environment name as an argument lists just that string (e.g. *env HOME* prints the HOME environment string).

Specifying an environment name followed by just an '=' deletes the environment string (e.g. *env FOOOPTS=* deletes the environment string for FOOOPTS). Specifying an environment name followed by an '=' and a value string defines the environment string for that name. Any existing string by that name is replaced by the new definition. (e.g. *env HOME= /public* sets the home directory to */public*. (Note: Changing HOME does not change the HOME/bin entry in the PATH environment.)

Multiple arguments are accepted. (e.g. *env HOME= /public TERM=* redefines the HOME string to be */public* and deletes the *TERM* environment string. If an environment string contains commas or spaces, the string must be quoted because commas and spaces are valid argument separators to the 4404 shell.

The C library functions *getenv, execve, execle, execlp,* and *execvp* are supported. The *execlp* and *execvp* routines use the PATH environment string to determine the search paths.

BACKUP

Page 2-3. Add the following options to the *SYNTAX* heading.

[+r][+T[=length]]

Page 2-4. Add to *Options Available* the following:

r — Retension streaming tape cartridge before any action. Using this option may avoid reading errors from the streaming tape drive. This option must be used in conjunction with the *+T* option.

T[=length] Backup to the streaming tape instead of floppy. The default parameter for the tape length is 450 feet. To backup to a 450 foot tape use *+T*; to backup to a 300 foot tape, use *+T=300.*

Page 2-8. Change the number of characters that a backup volume name may contain from *forty characters* to *126 characters.*

CC

Page 2-9. Under *SYNTAX,* add *[+p]* and *[+P].*

Under *Options Available* add the information that the following options cannot be combined; each must be preceeded by a space and "+" character and followed by a space.

Option *+D<name>[= <defn>]* must appear by itself on the command line.

Option *+i= <dir_name>* must appear as a separate option on the command line.

Option *+l= <lib_name>* must appear as a separate option on the command line.

Page 2-10. Under *Options Available,* add the following:

p Use stand alone pre-processor.

P Produce intermediate (.p) files and stop.

Add the following note at the bottom of page 2-10.

NOTE

The 'C' pre-processor is the file /bin/cpasses/cprep. If you want to use it with another program, it takes its input from stdin and writes its output to stdout.

CHD

Page 2-11. Add under *SEE ALSO,* the command *perms.*

COMMSET

Page 2-12. Under *Options Available* the baud rate specification for split baud rates should appear as *nnn/mmm.*

Page 2-13. Add the following options.

cts=disable Ignore RS-232C Clear-To-Send signal.

cts=enable Wait for Clear-To-Send true before transmitting.

CONSET

Page 2-17. Add the following options.

+crnl	Enable display <CR> as <CR><LF>.
-crnl	Disable display of <CR> as <CR><LF>.
default	Restore default settings.

COPY

Page 2-19. Add to the *SYNTAX* heading for directory copies (the second and third forms of the command) the options *+F* and *+M.*

Page 2-20. Add to *Options Available* the following:

D	Implicitly specify the high level directory names. This option works properly only in conjunction with the *+d* option. When used together with *+d*, *+D* preserves the source directory structure within the destination directory.

Change the discussion on the option *+P* to:

"Preserves all the characteristics of the file — the modification time and the owner."

Change the options on example 3 from *+los* to *+lo.*

DIR

Page 2-40. Add to the discussion of the *+t* option: "This option sorts *all* files in a directory by the time last modified. It cannot be used to sort specific files or groups of files (via wildcard characters)."

EDIT

Page 2-47. Change the maximum number of characters allowed by the operating system from *fourteen* to *55.*

ECHO

Page 2-46. Add to *SYNTAX* the argument *[+hex].*

Add to *Options Available* the line:

+hex	Send the equivalent hex byte to standard output.

FIND

Page 2-50. Add the following options to the *SYNTAX* section.

[+bns]

Page 2-51. Add the following to the *Options Available* section.

b Check files ending in *.bak*.

n Do not print line numbers on match.

s Print skipped file names.

FORMAT

Page 2-53. Add to *SYNTAX*: *[+n]*; add to *Options Available*:

n Do not issue the input prompt.

FREE

Page 2-55. Add to *Description*: "*free* also reports the disk usage (*blocks* and *fdns* used).

HEADSET

Page 2-58. Add the following options to the applicable descriptions on this page.

B Set or reset the "clear BSS memory" bit in the binary header...

d Set the *no core dump* bit in the binary header.

t Produce a shared-text executable module.

Change the <task_size> argument for *+b= <task_size>*. The acceptable arguments are 128K, 256k, 512K, 1M, 2M, 4M, or 8M.

LINK

Page 2-76. Remove the sentence that states that "Only the system manager may make a link to a directory."

LOAD

Page 2-80. Add *[+f] [+B] [+x]* to *SYNTAX.*

Change the sample file name in *Description* from */lib/std_env* to */lib/ldr_environ.*

Page 2-81. Add to option documentation:

B Set/clear BSS bit in binary header

Change the <task_size> argument for *+b= <task_size>*. The acceptable arguments are 128K, 256k, 512K, 1M, 2M, 4M, or 8M.

Page 2-83. Add to option documentation:

x= <file_name> Incremental load <file_name>

f In this format, text pages are loaded into the user's address space when first referenced (through page faulting) rather than at *exec* time.

LOGIN

Page 2-86. Remove the note that references the file *.home?*. This file no longer exists.

REMOVE

Page 2-104. Add *[+q]* to *SYNTAX.*

For the *+q* option, change the comment to read:

Quiet mode. Do not report any errors.

RENAME

Remove the error message:

File "<file_name>" is a directory

Remove the text that states "rename cannot rename directories".

RESTORE

Page 2-109. Remove the *[+a= days]* option from *SYNTAX.* Remove the *a= <days>* discussion from *Options Available.*

Add *[+r]* and *[+T[= L]]* to *SYNTAX.*

Page 110. Add the following to *Options Available.*

r — Retension streaming tape cartridge before any action. Using this option may avoud reading errors from the streaming tape drive. This option must be used in conjunction with the *+T* option.

T[=L] — Restore from streaming tape instead of floppy. The tape length parameter *L* default is 450 foot tape, eg. (+T=300 for 300 foot tape).

Add the following note to the *Options Available.*

NOTE

The "+d" option to restore entire directories creates subdirectories only if the original "backup" command specified "/" or "." as the directory to back up. Absolute sub-directories will ***not*** *be created, although the files contained within them will be restored if the subdirectory already exists. That is, the command*

backup . +dl

will save all subdirectories under the current working directory, and the command

restore +dl

will restore these subdirectories and their contents. However, the command

backup +dl /dir1/subdir2

while it saves the subdirectory "/dir1/subdir2" and its contents (including subsequent subdirectories) in absolute format, the command

restore +dl

will fail if any of the directories or subdirectories do not exist. The error messages are specific enough to allow you to manually create the directory structure necessary for "restore" to work. For an example of how this is used to control directory structure, see the script file "restoreFiles" on the "SYSINSTALL" diskette.

Remove the option *R* from *EXAMPLES* 1 and 2.

SHELL

Page 2-136. Add the command ***unset*** to the table of ***shell*** commands. ***unset*** removes named environment variables declared by ***set***. Use the option ***+a*** to remove all environment variables.

Page 2-136. Add to ***unalias*** the option ***+a***. This option removes all aliases.

TAIL

Change the description of *tail* on the top of page 2-145 from "(up to 250)" to "(default of 250)".

DEVCHECK

Page 3-7. Change the *Options* to read:

f Check only the fdn space.

s Check only the swap space.

v Check only the volume space.

MOUNT

Page 3-22. Add to *DESCRIPTION*:

The *mount* command with no arguments prints the status of any currently mounted devices.

SIGNALS

The following table lists the signals implemented on the 4404. Change the references on pages 2-66, 2-67, 2-68, 4-39, 4-40, 6-11, 7-62, 7-63, 7-97, and 7-98 to reflect the currently implimented signals. The operating system signals are defined in the file */lib/include/sys/signal.h.*

The possible actions (shown by "+") are:

Abort Default state is "abort" (otherwise "ignore")

Catch Interrupt can be caught

Dump Produces a core dump

Ignore Interrupt can be ignored

Reset Resets to default state when triggered

4404 Operating System Signals

Name	Number	Description	Abort	Catch	Dump	Ignore	Reset
SIGHUP	1	Hangup	+	+	-	+	+
SIGINT	2	Keyboard	+	+	-	+	+
SIGQUIT	3	Quit	+	+	+	+	+
SIGEMT	4	EMT $Axxx emulation	+	+	+	+	+
SIGKILL	5	Task kill (cannot be caught or ignored)	+	-	-	-	+
SIGPIPE	6	Broken pipe	+	+	-	+	+
SIGSWAP	7	Swap error	+	-	-	-	+
SIGTRACE	8	Trace	+	+	-	+	-
SIGALARM	10	Alarm	+	+	-	+	+
SIGTERM	11	Task terminate	+	+	-	+	+
SIGTRAPV	12	TRAPV instruction	+	+	+	+	+
SIGCHK	13	CHK instruction	+	+	+	+	+
SIGEMT2	14	EMT $Fxxx instruction	+	+	+	+	+
SIGTRAP1	15	TRAP #1 instruction	+	+	+	+	+
SIGTRAP2	16	TRAP #2 instruction	+	+	+	+	+
SIGTRAP3	17	TRAP #3 instruction	+	+	+	+	+
SIGTRAP4	18	TRAP #4 instruction	+	+	+	+	+
SIGTRAP5	19	TRAP #5 instruction	+	+	+	+	+
SIGTRAP6	20	TRAP #6 instruction	+	+	+	+	+
SIGPAR	21	Parity error	+	-	+	-	+
SIGILL	22	Illegal instruction	+	-	+	-	+
SIGDIV	23	DIVIDE by 0	+	+	+	+	+
SIGPRIV	24	Privileged instruction	+	-	+	-	+
SIGADDR	25	Address error	+	-	+	-	+
SIGDEAD*	26	Dead child	-	+	-	+	+
SIGWRIT	27	Write to READ-ONLY memory	+	-	+	-	+
SIGEXEC	28	Execute from STACK/DATA space	+	-	+	-	+
SIGBND	29	Segmentation violation	+	+	+	-	+
SIGUSR1	30	User-defined interrupt #1	+	+	-	+	+
SIGUSR2	31	User-defined interrupt #2	+	+	-	+	+
SIGUSR3	32	User-defined interrupt #3	+	+	-	+	+
SIGABORT	33	Program abort	+	-	+	-	+
SIGSPLR	34	Spooler interrupt	+	+	-	+	+
SIGINPUT	35	Input is ready	+	+	-	+	+
SIGDUMP	36	Memory dump	+	+	+	+	+
SIGMILLI	62	Millisecond alarm	+	+	-	+	+
SIGEVT	63	Mouse/keyboard event interrupt	+	+	-	+	+

* The operating system does not reset the signalling mechanism after once set (i.e. with an cpint(SIGDEAD,addr) call). The parent task must reset with an cpint(SIGDEAD,addr) call.

PHYS

Page 6-23. Remove objects 2 and 3 from the resources represented by "object".

FILE STATUS BUFFER TABLE

(Correction) The file status buffer table on page 4-26 contains incorrect field names. The proper names (from */lib/include/sys/stat.h*) are:

```
short   st_dev      /* device number */
short   st_ino      /* fdn number */
char    st_filler
char    st_mode     /* file mode (type) */
char    st_perm     /* file access permissions */
char    st_nlink    /* file link count */
short   st_uid      /* file owner's user id */
long    st_size     /* file size in bytes */
long    st_mtime    /* last modified time */
long    st_spr      /* spare - future use only */
```

'C' COMPILER

Several new features are available with the 'C' compiler. These are:

- Support for IEEE format floating-point.
- Several of the libraries have been changed and reorganized. See the entries in the */lib* directory for what is available. Most of the internal reorganization places internal modules as smaller sub-libraries.
- The */lib/graphics* library has been updated to include long procedure and variable names (Note: both the previous short form and the new long form exist with in the library).
  example:

  old... ClearScreen was actually represented as ClearSc

  now... is both ClearScreen and ClearSc

  There are three exceptions however: *FormDes*, *FormCre* and *Restore* no longer exist in the short (trucated) form. Instead they are available only as the long form *FormDestroy*, *FormCreate*, and *RestoreDisplayState*.
- The stand-alone *C* pre-processor is now available as *cprep* and supported through options *p* & *P* switches to the compiler (for pre-processed files with *.p* suffix.
- New libraries *popen* and *pclose* and support for opening files for update (ie. *r+*, *w+* and *a+*) through *fopen*, *freopen* and *fdopen*.
- New routines *addmount(device,directory)* adds an entry to the mount table (/etc/mtab); and *rmvmount(device)* removes an entry from the mount table (/etc/mtab).

- New routine *getppid()*, gets a parent process id.
- The C library functions *getenv, execve, execle, execlp,* and *execvp* offer environmental support. The *execlp* and *execvp* routines use the *PATH* environment string to determine the search paths.

  getenv() getenv searches the environmental list for a string form *name_value* and returns a pointer to the string *value* if such a string is present, otherwise getenv returns a null pointer (pointer of 0).

  execve() execve(name,argv,envp) execute a file.

  execle() execle(name,arg0, arg1,...,argn,0,envp) execute a file.

C SYSTEM AND LIBRARY CALLS

Some changes have been made in the 'C' library calls. These are:

phys

Page 7-83. Remove meanings 2 and 3 for the absolute value of <code>. These meanings are not available.

fprintf

Add the following description of *fprintf* to section 7 of the 4404 Reference Manual.

FPRINTF

Write formatted data to a file.

SYNOPSIS

```
#include <stdio.h>
int fprintf(stream, format [,arglist]
FILE *stream;
```

Arguments

stream — A pointer to the FILE into whitch the output is to be written.

format — A pointer to the format string to use for output conversion.

Returns

The number of characters written to ***stream*** or EOF if an error occurs.

DESCRIPTION

This function generates characters from the format description contained in the format string pointed to by *format* and the argument list *arglist* (if present) and writes these characters to the FILE *stream*. It returns the number of characters written to *stream*.

Format

The *format string* consists of two types of characters: plain characters that are printed just as they appear and conversion specifiers that cause successive arguments from the optional arglist to be converted and printed in their place.

Each conversion specification is introduced by the character %. Following the % character, there may be, in this order:

An optional minus sign (-). The minus sign negates signed numeric fields, causes unsigned numeric fields to be printed in (ones compliment?), and has no effect on alphanumeric fields.

n[...] An optional digit string specifying a field width. If the converted value has fewer characters than the field width, it is not normally padded unless the second width string is present.

An optional dot (.) that serves to separate the field width string from the next digit string.

n[...] An optional digit string that specifies a precision to which floating point numbers are to be printed, or the maximum number of characters of fixed-point numbers to print. This string will cause numeric strings that do not fill a field to be padded with zeros.

char A character that indicates the type of conversion to be perfomed.

Conversion Characters

d The corresponding integer argument from ***arglist*** will be converted to decimal notation.

o The corresponding integer argument from ***arglist*** will be converted to octal notation.

x The corresponding integer argument from ***arglist*** will be converted to hexadecimal notation.

f The corresponding float or double argument is converted to floating point notation in the style [-]d...d.d...d, where the whole number is before the decimal point and the number after is equal to the precision specification for the argument. Default precision specification is six digits.

e The corresponding float or double argument is converted to scientific notation in the style [-]d.ddd e (+/-) dd, where there is one digit before the decimal point and the number after is equal to the precision specification for the argument. Default precision specification is six digits.

g The float or double argument is printed in style *d*, *f*, or *e* — the choice is for the style giving full precision in minimum space.

c The single character argument is printed.

s *s* must be a pointer to a string. Characters from this string are printed until either a null (value 0) or the maximum number of characters indicated by the precision specification is reached. If the precision specification is missing, all characters up to a null are printed.

u The unsigned integer argument is converted to decimal and printed. The result will be in the range 0 through the maximum value that can be stored as an unsigned integer.

% Print a literal % — no argument conversion is done.

EXAMPLES

The following table shows what effect the format string has on the value printed by *fprintf*. fprintf(file,s, n); yields the following results:

Format String Examples

Value	Format String	Result Printed
Integer 1000	"%d"	1000
Integer 1000	"%-d"	-1000
Integer 1000	"%7d"	1000
Integer 1000	"%7.7d	0001000
Unsigned 1000	"%u"	1000
Unsigned 1000	"%-u"	4294966296
Float 100.00	"%f"	100.000000
Float 100.00	"%-f"	-100.000000
Float 100.00	"%7f"	100.000000
Float 100.00	"%7.1f"	100.0
Float 100.00	"%-7.1f"	-100.0
Float 100.00	"%7.7f"	100.0000000

mknod

Page 7-74. The bit strings for directories and block-special files are reversed. They should be:

0x0800	Make the file a directory
0x0200	Make the file a block-special file

NOTE

"mknod()" is a privileged call. You must be user "system to use it.

NOTE

"mknod()" does not create and link the "." and ".." files if you use it to create a directory. You must use the "link()" call to link "." to the directory itself and ".." to its parent.

mknod Example

The following two files, mknod.example and exMknod.c show how to use the mknod() call to create a directory. *mknode.example* is a *script* file (set the permissions to "execute"), while *exMknod.c* contains the 'C' code. You must be logged in as user *system* to execute this example, or you will get a "Permission denied" error.

File mknod.example

```
echo "= Example of mknod to make a directory ="
echo
echo "= cc exMknod.c ="
cc exMknod.c
echo
echo "= exMknod ="
exMknod
remove exMknod +l
echo
echo "   End mknod example"
```

File exMknod.c

```
/* This call is an example of how to create a new directory.
 * The name of the new directory will be 'fileName' and will
 * have the type and permission as stated in 'fileMode'. */

#include    <errno.h>
#include    <stdio.h>

extern    int   errno;

main()
{
   char   *fileName, *permString, *dotName;
   int    fileMode, permCode;

   printf("*** Begin mknod example ***0);

   permCode = 0x0007 + 0x0038;              /* rwx... + ...rwx */
   fileMode = 0x0800 + permCode;            /* directory + perms */
   fileName = "aDirectory";
   permString = permLabel(permCode);

   if (mknod(fileName,fileMode,0) != -1) {  /* create directory node */
      printf("PASSED: directory %s created with mode %d (%s)0,
             fileName, permCode, permString);

      dotName = "aDirectory/.";             /* link directory self */
      if (link(fileName,dotName) != -1)
         printf("PASSED: %s linked to %s0, dotName, fileName);
      else {
         printf("FAILED*: %s NOT linked to %s", dotName, fileName);
         perror(errno);
      }

      dotName = "aDirectory/..";            /* link directory parent */
      fileName = ".";                       /* link directory parent */
      if (link(fileName,dotName) != -1) {
         printf("PASSED: %s linked to %s0, dotName, fileName);
         fflush(stdout);
         system("dir +al aDirectory");

         fileName = "aDirectory";           /* unlink directory (remove) */
         if (unlink(fileName) != -1)
            printf("PASSED: %s unlinked (removed)0, fileName);
         else {
            printf("FAILED*: %s NOT unlinked (removed)", fileName);
            perror(errno);
         }
```

```
      }
      else {
         printf("FAILED*: %s NOT linked to %s", dotName, fileName);
         perror(errno);
      }
   }
   else {
      printf("FAILED*: directory %s NOT created with mode %d (%s)0,
            fileName, permCode, permString);
      perror(errno);
   }

   printf("*** End mknod example ***0);
}

/*
 * Set alphabetic label for permission code permCode.
 *
 * permLabel (permCode)
 */

char *permLabel(permCode)
int permCode;

{
   switch (permCode) {

      case  0: return "      ";
      case  1: return "r     ";
      case  2: return " w    ";
      case  3: return "rw    ";
      case  4: return "  x   ";
      case  5: return "r x   ";
      case  6: return " wx   ";
      case  7: return "rwx   ";
      case  8: return "   r  ";
      case  9: return "r  r  ";
      case 10: return " w r  ";
      case 11: return "rw r  ";
      case 12: return "  xr  ";
      case 13: return "r xr  ";
      case 14: return " wxr  ";
      case 15: return "rwxr  ";
      case 16: return "    w ";
      case 17: return "r   w ";
      case 18: return " w  w ";
      case 19: return "rw  w ";
      case 20: return "  x w ";
```

```
        case 21: return "r x w ";
        case 22: return " wx w ";
        case 23: return "rwx w ";
        case 24: return "   rw ";
        case 25: return "r  rw ";
        case 26: return " w rw ";
        case 27: return "rw rw ";
        case 28: return "  xrw ";
        case 29: return "r xrw ";
        case 30: return " wxrw ";
        case 31: return "rwxrw ";
        case 32: return "     x";
        case 33: return "r    x";
        case 34: return " w   x";
        case 35: return "rw   x";
        case 36: return "  x  x";
        case 37: return "r x  x";
        case 38: return " wx  x";
        case 39: return "rwx  x";
        case 40: return "   r x";
        case 41: return "r  r x";
        case 42: return " w r x";
        case 43: return "rw r x";
        case 44: return "  xr x";
        case 45: return "r xr x";
        case 46: return " wxr x";
        case 47: return "rwxr x";
        case 48: return "    wx";
        case 49: return "r   wx";
        case 50: return " w  wx";
        case 51: return "rw  wx";
        case 52: return "  x wx";
        case 53: return "r x wx";
        case 54: return " wx wx";
        case 55: return "rwx wx";
        case 56: return "   rwx";
        case 57: return "r  rwx";
        case 58: return " w rwx";
        case 59: return "rw rwx";
        case 60: return "  xrwx";
        case 61: return "r xrwx";
        case 62: return " wxrwx";
        case 63: return "rwxrwx";

        default: printf(" Invalid permission code.0);
                 return NULL;
    }
}
```

mount

Page 7-76. The description of the *rwflag* is reversed. If "rwflag" is zero, then the mount is for read-only. A non-zero value mounts the device for both read and write.

gtty, stty

Pages 7-56 through 7-60, 7-105 through 7-108. Some of the tty information fields referenced by *stty* and *gtty* are not available or ignored. See the files */lib/include/sys/sgtty.h* and */lib/systty* for these definitions.

The terminal modes that are ignored are: LCASE and CNTRL.

The protocol byte definitions that are ignored are: TRANS, IXON, and BAUD_RATE.

All the delay types; DELNL, DELCR, DELTB, DELVT, and DELFF are not available.

4404 USER'S MANUAL ADDITIONS AND ERRATA

INTRODUCTION

Version 1.5 of the operating system adds support to the Environment Variables, partitions the standard system software slightly differently, and a new command (for restoring the Smalltalk-80 demo files) has been added to the system rebuild tools. In addition, the following discussion of the script and shell commands should help to clear up any confusion as two how the two can be used.

SHELL AND SCRIPT

The 4404 operating system has two separate command interpreters, *script* and *shell.*

Both interpreters may be used interactively as well as in batch mode (meaning from command files). Each has some unique features, while other capabilities are common to both. The following table summarizes the features that are either unique to each interpreter or that have the same functionality with different syntax. See the 4404 Reference Manual for individual discussions of these commands.

shell and ***script*** Comparison

Feature	interactive *shell*	batch *shell*	interactive *script*	batch *script*
command and argument delimiters	<space>	<space>	<,> <space>	<,> <space>
sub-process termination	exit	exit EOF	log EOF	exit error with sabort on
background task limit	up to system limit	up to system limit	5	5
separators for multiple commands on a line	; &	; &	; || && ()	;|| && ()
stderr redirection	^	^	^ %	^ %
stderr append redirection	^^	^^	no	no
line continuation character	no	no	\	\
argument designators	$0 $1 $2 etc. $^ $$ $x-y $-y $* $n* $n- $-	$0 $1 $2 etc. $^ $$ $x-y $-y $* $n* $n- $-	$0 $1 $2 etc.	$0 $1 $2 etc.
command file flow control	no	no	sabort/proceed	sabort/proceed
command line editing	^P ^F ^B ^D ^H <Del> ^W ^A ^E ^K ^T ^U ^L ^Q Esc-F Esc-B Esc-D Esc-H	N/A	rub-out	N/A
History	^P ^N	N/A	no	N/A
Alias	yes	yes	no	no
Environment Variables	yes var = value	yes var = value	yes evn var = value	yes evn var = value
character quoting	\ " '	\ " '	' " \	' " \

The times when ***shell*** is executing .vs. the times when ***script*** is executing may not be altogether obvious to the user. This discussion makes these differences clear.

When first logged in, the user is executing an interactive ***shell***, called the "login shell". If the ***shell*** command is given with no arguments, the user will be put in an interactive ***shell*** child process. If the ***script*** command is given with no arguments, the user will be put in an interactive ***script*** child process.

When dealing with command files, there are two types to consider. Those with execute permission, and those without. Those without must be executed with either the ***shell*** or ***script*** command. Those with may be executed with the ***shell*** or ***script*** command, or simply by typing the file name.

When executing a command file with the *script* command, the *script* interperter will always process the file. Executing a command file with execute permissions by just giving the file name will cause the file to be processed by *script*.

When executing a command file with the *shell* command, the *shell* interpeter will process the file only if the *+i* option is given. Executing a command file, with the *shell* command without the *+i* option will cause the file to be processed by *script*.

Some shell Features with Regard to shell Processes

The PATH, alias, environment variables, and history features are available within any interactive *shell* process. Within a batch shell process, only PATH, alias, and environment variables are available; history is not (history is an interactive-only feature). Whenever a new shellprocess is begun these values are read from the file /<home-dir>/.shellhistory . This file is updated by the system whenever a logout occurs, or when an interactive shell process is exited. Therefore, a new shell-process will obtain it's environment from the most recent /<home-dir>/.shellhistory update.

4404 User's Manual Changes

Please make the following changes in your 4404 User's Manual.

The 4404 Mouse

Page 1-9. The 4404 is now shipped with a mechanical, rather than an optical, mouse. Delete any references to "mouse pad" in this topic. The mechanical mouse detects motion by sensing the motion of an internal rubber-coated ball rolled over any smooth surface.

Section 5, Recovery and Rebuild

Page 5-5, 5-18, 5-19. The smalltalk system files now consist of only the standard system, image, and changes files. The smalltalk demo files are now a separate group. The command to restore the smalltalk demo files (during system rebuild) is *restoreSTD*.

V1.5 Operating System Task Size Changes.

Operating system support for task sizes has changed. The new program file header format is different so, programs such as *EMACS, Smalltalk-80 (interpreter), LISP, Mprolog,* etc. require that "headset" be executed on them in order that the previous process size be reset. An old process size of 8 megabytes now maps to a process size of 1 megabytes.

The old task sizes were: 128K (defualt), 512K, 2048K, 8192K, 2M or 8M.

The new task sizes are: 128K (defualt), 256K, 512K, 1M, 2M, 4M or 8M.

EXAMPLE

In order to set an 8 mega-byte process size for Smalltalk-80, do the following:

login system headset /bin/smalltalk +b=8M

The *load* utility *headset* and other utilities have been updated accordingly. Old executable files if different from the 128K default task size, must have their task size reset to the desired size by invoking *headset.* The optional software files affected are as follows:

emacs: /bin/emacs +b=8M

franzLisp: /bin/lisp +b=8M
/bin/liszt +b=8M
/bin/lxref +b=128k (default)
/bin/lib/as +b=2M
(and user created
executable lisp
programs also)

/mprolog: /bin/mpro +b=1M (minimum)
/bin/prtr +b=1M (minimum)
/bin/cons +b=1M (minimum)