68NW9209H46A

SYSTEM V/88 Release 3.2 POSIX Conformance Guide

MOTOROL

Motorola welcomes your c

anual Title _____

art Number _____

Your Name _____

Your Title _____

Company _____

Address _____

_____

_____

General Information:

Do you read this manu

☐ Install the product

☐ Reference inforr

In general, how do yo

☐ Index ☐ Table

mpletenessl

hat topic would you

_____

_____

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 2565 PHOENIX, ARIZONA

POSTAGE WILL BE PAID BY ADDRESSEE

MOTOROLA INC.

Computer Group, Microcomputer Division, DW164

2900 South Diablo Way

Tempe, AZ 85282-9741

Presentation: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

What features of the manual are most useful (tables, figures, appendixes, index, etc.)?

Is the information easy to understand? ☐ Yes ☐ No If you checked no, please expla

Is the information easy to find? ☐ Yes ☐ No If you checked no, please explain:

Technical Accuracy: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

If you have found technical or typographical errors, please list them here.

Page Number	Description of Error

SYSTEM V/88 Release 3.2

POSIX Conformance Guide

(68NW9209H46A)

First Edition February 1990

Contents

1. Scope

This document describes conformance to IEEE Std 1003.1-1988 as required by section 2.2.1.2 of that standard for the implementation. Conformance type is IEEE Std 1003.1-1988, C Language Binding (Common Usage C Language-Dependent System Support).

It is assumed that the standard is available for reference. This document describes only the behavior that is specified to be implementation defined or where it is specified that behavior of implementations may vary.

This document has the same structure as the standard with the information presented in the appropriately numbered sections. Section titles without text indicate exact conformance to the standard. All descriptions indicate Motorola implementations of the standard.

Since Federal Information Processing Standard (FIPS) for POSIX is based on IEEE Std 1003.1-1988 with some additional requirements, conformance to IEEE Std 1003.1-1988 generally indicates conformance to the POSIX FIPS.

2. Definitions and General Requirements

2.1 Terminology

2.2 Conformance

2.2.1 Implementation Conformance

2.2.2 Application Conformance

2.2.3 Language-Dependent Services for the C Programming Language

2.2.3.1 Types of Conformance

Conformance type is IEEE Std 1003.1-1988, C Language Binding (Common Usage C Language-Dependent System Support).

2.2.3.2 C Standard Language-Dependent System Support

2.2.3.3 Common Usage C Language-Dependent System Support

2.2.4 Other C Language Related Specifications

2.3 General Terms

If a function call requires **appropriate privileges**, the effective user ID of the calling process must be zero.

In addition to the **file** types listed in this standard, *symbolic link*, *multiplexed character special*, and *multiplexed block special* types are supported. No other members of **file group class** are supported.

If a **pathname** begins with two successive *slashes*, the two are interpreted as a single *slash*.

A **read only file system** is a file system that prohibits any modification of the file system; modifications to superblock, creation or modification of inodes or files are not permitted.

The **system process** has *process ID* 1 and is an instance of the command **/etc/init**. The *system process* inherits *child processes* "orphaned" by the death of their *parent process*.

2.4 General Concepts

There are no **extended security controls**.

There are no additional or alternative **file access permissions** or file access control mechanisms.

File time update fields marked for update are updated by the next system update from a **sync**() system call, a **uadmin()** system call given the A_SHUTDOWN *cmd*, and periodically by the kernel (the period is a kernel configuration parameter). The fields for individual files is updated immediately by a **write**() or **writev**() if the O_SYNC file status flag is set, and by the **stat**(), **fstat**(), **rename**(), **utime**() system calls. The O_SYNC file status flag can be set by an **open**() or **fcntl**() system call.

2.5 Error Numbers

The following are error numbers defined in <errno.h>:

```
#define     EPERM       1    /* Not super-user                  */
#define     ENOENT      2    /* No such file or directory       */
#define     ESRCH       3    /* No such process                 */
#define     EINTR       4    /* interrupted system call         */
#define     EIO         5    /* I/O error                       */
#define     ENXIO       6    /* No such device or address       */
#define     E2BIG       7    /* Arg list too long               */
#define     ENOEXEC     8    /* Exec format error               */
#define     EBADF       9    /* Bad file number                 */
#define     ECHILD      10   /* No children                     */
#define     EAGAIN      11   /* No more processes               */
#define     ENOMEM      12   /* Not enough core                 */
#define     EACCES      13   /* Permission denied               */
#define     EFAULT      14   /* Bad address                     */
#define     ENOTBLK     15   /* Block device required           */
```

```
#define     EBUSY       16   /* Mount device busy                  */
#define     EEXIST      17   /* File exists                        */
#define     EXDEV       18   /* Cross-device link                  */
#define     ENODEV      19   /* No such device                     */
#define     ENOTDIR     20   /* Not a directory                    */
#define     EISDIR      21   /* Is a directory                     */
#define     EINVAL      22   /* Invalid argument                   */
#define     ENFILE      23   /* File table overflow                */
#define     EMFILE      24   /* Too many open files                */
#define     ENOTTY      25   /* Not a typewriter                   */
#define     ETXTBSY     26   /* Text file busy                     */
#define     EFBIG       27   /* File too large                     */
#define     ENOSPC      28   /* No space left on device            */
#define     ESPIPE      29   /* Illegal seek                       */
#define     EROFS       30   /* Read only file system              */
#define     EMLINK      31   /* Too many links                     */
#define     EPIPE       32   /* Broken pipe                        */
#define     EDOM        33   /* Math arg out of domain of func     */
#define     ERANGE      34   /* Math result not representable      */
#define     ENOMSG      35   /* No message of desired type         */
#define     EIDRM       36   /* Identifier removed                 */
#define     ECHRNG      37   /* Channel number out of range        */
#define     EL2NSYNC    38   /* Level 2 not synchronized           */
#define     EL3HLT      39   /* Level 3 halted                     */
#define     EL3RST      40   /* Level 3 reset                      */
#define     ELNRNG      41   /* Link number out of range           */
#define     EUNATCH     42   /* Protocol driver not attached       */
#define     ENOCSI      43   /* No CSI structure available         */
#define     EL2HLT      44   /* Level 2 halted                     */
#define     EDEADLK     45   /* Deadlock condition.                */
#define     ENOLCK      46   /* No record locks available.         */
```

*/Convergent Error Returns */

```
#define   EBADE           50 /* invalid exchange                    */
#define   EBADR           51 /* invalid request descriptor          */
#define   EXFULL          52 /* exchange full                       */
#define   ENOANO          53 /* no anode                            */
#define   EBADRQC         54 /* invalid request code                */
#define   EBADSLT         55 /* invalid slot                        */
#define   EDEADLOCK       56 /* file locking deadlock error         */
#define   EBFONT          57 /* bad font file fmt                   */
```

/* Stream Problems */

```
#define   ENOSTR          60 /* Device not a stream                 */
#define   ENODATA         61 /* no data (for no delay io)           */
#define   ETIME           62 /* timer expired                       */
#define   ENOSR           63 /* out of streams resources            */
#define   ENONET          64 /* Machine is not on the network       */
#define   ENOPKG          65 /* Package not installed               */
#define   EREMOTE         66 /* The object is remote                */
#define   ENOLINK         67 /* the link has been severed           */
#define   EADV            68 /* advertise error                     */
#define   ESRMNT          69 /* srmount error                       */
#define   ECOMM           70 /* Communication error on send         */
#define   EPROTO          71 /* Protocol error                      */
#define   EMULTIHOP       74 /* multihop attempted                  */
#define   EDOTDOT         76 /* Cross mount point (not really error) */
#define   EBADMSG         77 /* trying to read unreadable message   */
#define   ENAMETOOLONG    78 /* Filename too long                   */
#define   ENOTUNIQ        80 /* given log. name not unique          */
#define   EBADFD          81 /* f.d. invalid for this operation     */
#define   EREMCHG         82 /* Remote address changed              */
```

/* Shared Library Problems */

```
#define   ELIBACC          83      /* Can't access a needed shared lib.    */
#define   ELIBBAD          84      /* Accessing a corrupted shared lib.    */
#define   ELIBSCN          85      /* .lib section in a.out corrupted.     */
#define   ELIBMAX          86      /* Attempting to link in too many libs. */
#define   ELIBEXEC         87      /* Attempting to exec a shared library. */
#define   ENOSYS           89      /* unsupported system call (BCS 8.2)    */
#define   ELOOP            90      /* Too many levels of symbolic links    */
#define   ERESTART         91      /* BCS                                  */
```

/* Socket/network Error Returns*/

```
#define   EWOULDBLOCK      EDEADLK /* Operation would block                */
#define   EINPROGRESS      128     /* Operation now in progress            */
#define   EALREADY         129     /* Operation already in progress        */
#define   ENOTSOCK         130     /* Socket operation on non-socket       */
#define   EDESTADDRREQ     131     /* Destination address required         */
#define   EMSGSIZE         132     /* Message too long                     */
#define   EPROTOTYPE       133     /* Protocol wrong type for socket       */
#define   ENOPROTOOPT      134     /* Protocol not available               */
#define   EPROTONOSUPPORT  135     /* Protocol not supported               */
#define   ESOCKTNOSUPPORT  136     /* Socket type not supported            */
#define   EOPNOTSUPP       137     /* Operation not supported on socket    */
#define   EPFNOSUPPORT     138     /* Protocol family not supported        */
#define   EAFNOSUPPORT     139     /* Proto fam doesn't support addr fam   */
#define   EADDRINUSE       140     /* Address already in use               */
#define   EADDRNOTAVAIL    141     /* Can't assign requested address       */
#define   ENETDOWN         142     /* Network is down                      */
#define   ENETUNREACH      143     /* Network is unreachable               */
#define   ENETRESET        144     /* Network dropped connection on reset  */
#define   ECONNABORTED     145     /* Software caused connection abort     */
#define   ECONNRESET       146     /* Connection reset by peer             */
#define   ENOBUFS          147     /* No buffer space available            */
#define   EISCONN          148     /* Socket is already connected          */
#define   ENOTCONN         149     /* Socket is not connected              */
#define   ESHUTDOWN        150     /* Can't send after socket shutdown     */
#define   ETOOMANYREFS     151     /* Too many references: can't splice    */
#define   ETIMEDOUT        152     /* Connection timed out                 */
#define   ECONNREFUSED     153     /* Connection refused                   */
```

/* Additional POSIX/BCS errors */

```
#define     ENOTEMPTY       158    /* Directory not empty   */
```

/* End of POSIX/BCS defined errors */

The error [EFAULT] is reliably detected and the value is used when the system detects an invalid address in an attempt to use an argument of a call.

EFBIG occurs when the maximum file size is exceeded. The maximum file size for a process is configurable using **ulimit**(2). The system-imposed write limit for the maximum file size of a process is a system (kernel) configuration parameter, CDLIMIT. A field in the kernel *var* structure is initialized to CDLIMIT and is distributed with the value 32768 in 512-byte units (16Mb).

2.6 Primitive System Data Types

The following are the types and access definitions in <**sys/types.h**>:

```
typedef     long            daddr_t;  /* <disk address> type          */
typedef     char *          caddr_t;  /* <core address> type          */
typedef     unsigned long   ino_t;    /* <inode> type                 */
typedef     short           cnt_t;    /* <count> type                 */
typedef     long            time_t;   /* <time> type                  */
#ifdef      m88k
#define     LABELSIZE       24
#else
#ifdef      m68k
#define     LABELSIZE       14
#else
#define     LABELSIZE       10
#endif      /* m68k */
#endif      /* m88k */
typedef     int             label_t[LABELSIZE];
typedef     unsigned long   dev_t;    /* <old device number> type     */
typedef     long            off_t;    /* <offset> type                */
typedef     long            paddr_t;  /* <physical address> type      */
typedef     long            key_t;    /* IPC key type                 */
typedef     unsigned char   use_t;    /* use count for swap.          */
typedef     short           sysid_t;
typedef     short           index_t;
typedef     short           lock_t;   /* lock work for busy wait      */
typedef     unsigned int    size_t;   /* len param for string funcs   */
typedef     unsigned long   clock_t;  /* from C standard (POSIX)      */
                                      /* (BCS 6 3.3)                  */
```

```
typedef  unsigned long   mode_t;     /* for file attrs (POSIX 12.2 2.6) */
                                     /* (BCS)                           */
typedef  unsigned long   uid_t;      /* for user IDs (POSIX 12.2 2.6)   */
                                     /* (BCS 6 3.3)                     */
typedef  long pid_t;                 /* for process IDs (POSIX 12.2 2.6)*/
                                     /* (BCS 6 3.3)                     */
typedef  unsigned long   gid_t;      /* for group IDs (POSIX 12.2 2.6)  */
                                     /* (BCS 6 3.3)                     */
typedef  long            nlink_t;    /* for link counts (POSIX 12.2 2.6)*/
```

```
/*
 * Distributed SYSTEM V/88 hook
 */
```

```
typedef struct cookie {
  long  c_sysid;
  long  c_rcvd;
} *cookie_t;
```

2.7 Environment Description

Characters other than those in the *portable filename character set* are permitted in environment variable *names.*

2.8 C Language Definitions

2.8.1 Symbols From The C Standard

2.8.2 POSIX Symbols

No additional feature test macros are defined.

2.8.3 Headers and Function Prototypes

2.9 Numerical Limits

The following are symbolic constants specifically undefined in <**limits.h**>. The values associated with these constants are, instead, obtained from calls to **sysconf**() or **pathconf**() at runtime.

```
#undef   ARG_MAX        /* traditional max for exec args                */
#undef   CHILD_MAX      /* max # of processes per user id               */
#undef   LINK_MAX       /* max # of links to a single file              */
#undef   NAME_MAX       /* max # of characters in a file name           */
#undef   OPEN_MAX       /* max # of files a process can                 */
                        /* have open                                    */
#undef   PATH_MAX       /* max # of characters in a path name           */
#undef   PIPE_BUF       /* max # bytes atomic in write to have open     */
#undef   MAX_CANON      /* maximum number of bytes in a                 */
                        /* terminal input line                          */
#undef   MAX_INPUT      /* Minimum number of bytes for which            */
                        /* space is guaranteed in                       */
                        /* a terminal input queue.                      */
```

The following are the symbolic constants defined in **<limits.h>**:

```
#define  CHAR_BIT    8           /* # of bits in a "char"                */
#define  CHAR_MAX    255         /* max integer value of a "char"        */
#define  CHAR_MIN    0           /* min integer value of a "char"        */
#define  DBL_DIG     16          /* digits of precision of a "double"    */
#define  DBL_MAX     1.79769313486231470e+308
                                 /* max decimal value of a "double"      */
#define  DBL_MIN     ((double)4.4501477170144023e-308)
                                 /* min decimal value of a "double"      */
#define  FCHR_MAX    1048576     /* max size of a file in bytes          */
#define  FLT_DIG     7           /* digits of precision of a "float"     */
#define  FLT_MAX     3.40282346638528860e+38
                                 /* max decimal value of a "float"       */
#define  FLT_MIN     1.40129846432481707e-45
                                 /* min decimal value of a "float"       */
#define  HUGE_VAL    FLT_MAX
                                 /* error value returned by math lib     */
#define  INT_MAX     2147483647
                                 /* max decimal value of an "int"        */
#define  INT_MIN     -2147483648
                                 /* min decimal value of an "int"        */
#define  UINT_MAX    4294967295
                                 /* max decimal value of an "unsigned"   */
#define  LONG_MAX    2147483647
                                 /* max decimal value of a "long"        */
#define  LONG_MIN    -2147483648
                                 /* min decimal value of a "long"        */
#define  ULONG_MAX   4294967295
                                 /* max decimal value of an "unsigned"   */
#define  PASS_MAX    8           /* max # of characters in a password    */
#define  PID_MAX     30000       /* max value for a process ID           */
```

```
#define  PIPE_MAX      8192        /* max # bytes written to a pipe in   */
                                   /* a write                            */
#define  SHRT_MAX      32767       /* max decimal value of a "short"     */
#define  SHRT_MIN      -32768      /* min decimal value of a "short"     */
#define  STD_BLK       1024        /* # bytes in a physical I/O block    */
#define  SYS_NMLN      9           /* # of chars in uname-returned       */
                                   /* strings                            */
#define  UID_MAX        60000      /* max value for a user or group ID   */
#define  USI_MAX       UINT_MAX
                                   /* max decimal value of an "unsigned" */
#define  WORD_BIT      32          /* # of bits in a "word" or "int"     */
#define  MB_LEN_MAX    1           /* max number of bytes in a           */
                                   /* multibyte character                */
#define  SCHAR_MAX     127         /* max value for signed char          */
#define  SCHAR_MIN     -127        /* min value for signed char          */
#define  UCHAR_MAX     255         /* max value for unsigned char        */
#define  NGROUPS_MAX   0           /* Minimum maximum number of          */
                                   /* simultaneous supplementary group   */
                                   /* IDs per process                    */
```

/*
 * POSIX limits - actual values for this system available from sysconf()
 */

```
#define  _POSIX_ARG_MAX     4096 /* length of the arguments for exec   */
                                 /* in bytes including environment     */
                                 /* data                               */
#define  _POSIX_CHILD_MAX   6    /* The number of simultaneous procs   */
                                 /* per user ID                        */
#define  _POSIX_LINK_MAX    8    /* The value of a file's link count   */
#define  _POSIX_MAX_CANON   255  /* The number of bytes in a terminal  */
                                 /* canonical input queue              */
#define  _POSIX_MAX_INPUT   255  /* The number of bytes for which      */
                                 /* space is guaranteed to be          */
                                 /* available for terminal             */
                                 /* input queue                         */
#define  _POSIX_NAME_MAX     14  /* Number of bytes in a filename      */
#define  _POSIX_NGROUPS_MAX  0   /* Number of simultaneous supp.       */
                                 /* group IDs per process              */
#define  _POSIX_OPEN_MAX     16  /* The number of files that one proc  */
                                 /* can have open at a time            */
#define  _POSIX_PATH_MAX    255  /* Max number of bytes in a pathname  */
#define  _POSIX_PIPE_BUF    512  /* Number of bytes guaranteed to be   */
                                 /* written atomically to a pipe       */
```

2.9.1 C Language Limits

2.9.2 Minimum Values

2.9.3 Run-Time Increasable Values

NGROUPS_MAX is defined in <**limits.h**> to be 0, but the actual value supported is provided at runtime by **sysconf**().

2.9.4 Run-Time Invariant Values (Possibly Indeterminate)

ARG_MAX, CHILD_MAX and OPEN_MAX are indeterminate; they are undefined in <**limits.h**>. The actual value associated with ARG_MAX is provided at runtime by **sysconf**() and is 8192. The actual values supported with OPEN_MAX and CHILD_MAX are provided by **sysconf**() and are kernel configuration parameters from the **master** file.

2.9.5 Pathname Variable Values

LINK_MAX, MAX_CANON, MAX_INPUT, NAME_MAX, PATH_MAX and PIPE_BUF are not defined in <**limits.h**>. The actual values supported for these are provided at runtime by **pathconf**().

2.10 Symbolic Constants

The following are the symbolic constants defined in <**unistd.h**>:

/* Symbolic constants for the "access" routine: */

```
#define  R_OK       4     /* Test for Read permission                */
#define  W_OK       2     /* Test for Write permission               */
#define  X_OK       1     /* Test for eXecute permission             */
#define  F_OK       0     /* Test for existence of File              */
#define  F_ULOCK    0     /* Unlock a previously locked region       */
#define  F_LOCK     1     /* Lock a region for exclusive use         */
#define  F_TLOCK    2     /* Test and lock a region for exclusive use */
#define  F_TEST     3     /* Test a region for other processes locks  */
```

```
/* Symbolic constants for the "lseek" routine: */
#define  SEEK_SET   0    /* Set file pointer to "offset"             */
#define  SEEK_CUR   1    /* Set file pointer to current plus "offset" */
#define  SEEK_END   2    /* Set file pointer to EOF plus "offset"    */

/* Path names */
#define GF_PATH "/etc/group" /* Path name of the "group" file        */
#define PF_PATH "/etc/passwd"/* Path name of the "passwd" file       */

/*
 * Following values are POSIX requirements
 */
#define  NULL                        0

/* POSIX option flags */
#define  _POSIX_JOB_CONTROL          1
#undef   _POSIX_CHOWN_RESTRICTED
#define  _POSIX_SAVED_IDS            1
#define  _POSIX_NO_TRUNC             1
#undef   _POSIX_VDISABLE
#define  _POSIX_VERSION              198808L

/* The following defines are specified in POSIX draft 12.0 and are therefore
 * necessary to compile the early NBS-PCTS
 */
#define  _POSIX_GROUP_PARENT         0
#define  _POSIX_CHOWN_SUP_GRP        0
#define  _POSIX_DIR_DOTS             0
#define  _POSIX_UTIME_OWNER          0

/* sysconf() names (POSIX and BCS) */
#define  _SC_ARG_MAX           1 /* Bytes allowed for exec arguments   */
#define  _SC_CHILD_MAX         2 /* Max child processes                */
#define  _SC_CLK_TCK           3 /* Clock tick rate (HZ)               */
#define  _SC_NGROUPS_MAX       4 /* Max multiple groups                */
#define  _SC_OPEN_MAX          5 /* Max open files                     */
#define  _SC_JOB_CONTROL       6 /* Job control support                */
#define  _SC_SAVED_IDS         7 /* saved-set-uid/gid support          */
#define  _SC_VERSION           8 /* Posix version stamp                */
#define  _SC_BCS_VERSION       9 /* BCS version stamp                  */
#define  _SC_BCS_VENDOR_STAMP 10 /* Vendor stamp of system             */
#define  _SC_BCS_SYS_ID       11 /* unique machine id                  */
#define  _SC_MAXUMEMV         12 /* Max user process size 1-KB pages   */
#define  _SC_MAXUPROC         13 /* Max number of processes/user       */
#define  _SC_MAXMSGSZ         14 /* Max size of a message              */
#define  _SC_NMSGHDRS         15 /* Total number of msg headers/system */
```

```
#define  _SC_SHMMAXSZ          16 /* Maximum size of shared segment     */
#define  _SC_SHMMINSZ          17 /* Minimum size of shared segment     */
#define  _SC_SHMSEGS           18 /* Max attached segs/process          */
#define  _SC_NMSYSSEM          19 /* Total number semaphores/system     */
#define  _SC_MAXSEMVL          20 /* Max semaphore value                */
#define  _SC_NSEMMAP           21 /* Number of semaphore sets           */
#define  _SC_NSEMMSL           22 /* Number of semaphores/set           */
#define  _SC_NSHMMNI           23 /* Number of shared segments/system   */
#define  _SC_ITIMER_VIRT       24 /* System supports virtual timer      */
#define  _SC_ITIMER_PROF       25 /* System supports profiling timer    */
#define  _SC_TIMER_GRAN        26 /* Granularity of timers in usec      */
#define  _SC_PHYSMEM           27 /* Total physical memory/system (kb)  */
#define  _SC_AVAILMEM          28 /* Total physmem avail to user (kb)   */
#define  _SC_NICE              29 /* nice prioritization is supported   */
#define  _SC_MEMCTL_UNIT       30 /* bytes in a memory unit             */
                                  /* in memctl system call              */
#define  _SC_SHMLBA            31 /* Memory address rounding used by    */
                                  /* shmsys in bytes                    */
#define  _SC_SVSTREAMS         32 /* System V streams are supported     */
#define  _SC_CPUID             33 /* return Processor Identification
                                     Register                           */
```

/* pathconf() names (POSIX and BCS) */

```
#define  _PC_LINK_MAX            1
#define  _PC_MAX_CANON           2
#define  _PC_MAX_INPUT           3
#define  _PC_NAME_MAX            4
#define  _PC_PATH_MAX            5
#define  _PC_PIPE_BUF            6
#define  _PC_CHOWN_RESTRICTED    7
#define  _PC_NO_TRUNC            8
#define  _PC_VDISABLE            9
#define  _PC_BLKSIZE            10
```

/* The following values are specified in POSIX 12.0 and
 * are therefore necessary to compile the NBS-PCTS
 */

```
#define  _PC_CHOWN_SUP_GRP      11
#define  _PC_DIR_DOTS           12
#define  _PC_GROUP_PARENT       13
#define  _PC_UTIME_OWNER        14
```

/* Symbolic support for BCS requirements */

```
#define  _BCS_VERSION          198902L
                                        /* _SC_BCS_VERSION number    */
#define  _BCS_ITIMER_VIRT      1        /* Virtual timer support     */
#define  _BCS_ITIMER_PROF      1        /* Profiling timer support   */
#define  _BCS_NICE             1        /* Nice priorization support */
#define  _BCS_SVSTREAMS        1        /* System V streams support  */
#define  _BCS_PTRACE_MAGIC     0x00088000 /* Ptrace_user magic number  */
#define  _BCS_PTRACE_REV       0x00000001 /* Ptrace_user version number*/
```

/* ulimit symbolic constants (BCS) */

```
#define  GET_ULIMIT            1
#define  SET_ULIMIT            2
#define  GET_BREAK             3
#define  GET_MAX_OPEN          4
```

3. Process Primitives

3.1 Process Creation and Execution

3.1.1 Process Creation

Function: **fork()**

The following additional process characteristics are inherited by the child process:

environment
close-on-exec flag
signal handling settings
set-user-ID mode bit
set-group-ID mode bit
profiling on/off status
nice value
all attached shared memory segments
process group ID
tty group ID
trace flag
current working directory
root directory
file mode creation mask
file size limit

The **semadj** values are cleared for the child process. Process locks, text locks and data locks are not inherited by the child.

3.1.2 Execute a File

Functions: **execl**(), **execv**(), **execle**(), **execve**(), **execlp**(), **execvp**()

In functions **execlp**() and **execvp**(), if the *file* argument does not contain a slash character and if the PATH environment variable is not present, the directories searched to find the file are **/bin** and **/usr/bin**.

The number of bytes available for a new process's argument and environment lists, as returned by **sysconf**(SC_ARG_MAX), includes null terminators.

The following additional process attributes are inherited by the child process image:

nice value
semadj values
tty group ID
trace flag
file size limit

Profiling is disabled in the new image.

3.2 Process Termination

3.2.1 Wait for Process Termination

Functions: **wait**(), **waitpid**()

If a child is stopped due to a trace breakpoint, **wait**() returns immediately. If the child process is stopped, the high order 8 bits of *status* contain the number of the signal that caused the process to stop; the low order 8 bits are set equal to 0177 (octal).

If a parent process terminates without waiting for its children to terminate, the children are assigned the parent process ID 1. This process ID corresponds to the initialization process, **/etc/init**.

3.2.2 Terminate a Process

Function: **_exit()**

If a parent process terminates without waiting for its children to terminate, the children are assigned the parent process ID 1. This process ID corresponds to the initialization process, **/etc/init**.

The SIGCHLD signal is supported.

Job control is supported.

3.3 Signals

3.3.1 Signal Concepts

3.3.1.1 Signal Names

All signals shown in Tables 3-1 and 3-2 are supported. The following additional signals may occur in the system:

SIGTRAP	trace trap
SIGIOT	IOT instruction
SIGEMT	EMT instruction
SIGBUS	bus error
SIGSYS	bad argument to a system call
SIGPWR	power-fail restart
SIGVTALRM	virtual time alarm
SIGPROF	profiling timer alarm
SIGWINCH	window size change
SIGURG	urgent condition present on socket
SIGPOLL	pollable event occurred

3.3.1.2 Signal Generation and Delivery

If there is a subsequent occurrence of a pending signal, the signal is delivered once.

In addition, on SYSTEM V/88, multiple exceptions may exist when a signal is taken. Using the signal frame format specified in <**signal.h**>, multiple instances of a particular type of exception generate a single signal as the result of entering the system with multiple exceptions. The information associated with each separate exception is placed on the users stack. The signal interface code linked into the user application may then parse this information and call the users signal handler, as appropriate.

Signals generated from a **kill**() system call are distinct from signals from a set of exceptions. Each is treated as the occurrence of one signal; **kill**() has zero *exception blocks*; exceptions have multiple *exception blocks.*

SIGTRAP is generated as a result of an exception due to trap on **execve**() system call (if tracing), or trap to vectors 504-511.

SIGIOT is not generated as a result of an exception.

SIGEMT is not generated as a result of an exception.

SIGBUS is generated as a result of an exception caused by a misaligned data access or a protection violation.

3.3.1.3 Signal Actions

By default, all signals except SIGCLD, SIGPWR, and SIGWINCH cause termination of the receiving process. SIGCLD, SIGPWR, and SIGWINCH are ignored by default. SIGPOLL can only be received by a process that has requested it.

3.3.2 Send a Signal to a Process

Function: **kill**()

3.3.2.2 Description

If a receiving process's effective user ID has been altered using the S_ISUID mode bit, the application receives a signal sent by the parent process or by a process with the same real user ID.

Since there are no extended security controls, there are no additional restrictions on sending a signal to a process.

If *pid* is zero, the *system processes* with process ID 0 and 1 do not receive the signal.

3.3.3 Manipulate Signal Sets

3.3.4 Examine and Change Signal Action

Function: **sigaction**() There are no additional flag bits for *sa_flags* in the *sigaction* structure defined in <**signal.h**>.

3.3.5 Examine and Change Blocked Signals

3.3.6 Examine Pending Signals

Function: **sigpending**()

3.3.6.4 Errors

If *set* points to an invalid address, **sigpending**() returns -1 and sets *errno* to EFAULT.

3.3.7 Wait for a Signal

3.4 Timer Operations

3.4.1 Schedule Alarm

3.4.2 Suspend Process Execution

3.4.3 Delay Process Execution

4. Process Environment

4.1 Process Identification

4.1.1 Get Process and Parent Process IDs

4.2 User Identification

4.2.1 Get Real User, Effective User, Real Group, and Effective Group IDs

4.2.2 Set User and Group IDs

4.2.3 Get Supplementary Group IDs

4.2.4 Get User Name

Functions: **getlogin()**, **cuserid()**

4.2.4.4 Errors

There are no error conditions or additional return values for **cuserid()** or **getlogin()**.

4.3 Process Groups

4.3.1 Get Process Group ID

4.3.2 Create Session and Set Process Group ID

4.3.3 Set Process Group ID for Job Control

Function: **setpgid()**

4.3.3.2 Description

The symbol {_POSIX_JOB_CONTROL} is defined and the **setpgid()** function is supported as described.

4.4 System Identification

4.4.1 System Name

Function: **uname()**

4.4.1.2 Description

The following is the **utsname** structure in **<sys/utsname.h>**:

```
struct utsname {
        char    sysname [256];
        char    nodename[256];
        char    release [256];
        char    version [256];
        char    machine [256];
};
```

The values for members of *utsname* are constants defined at the time the system is installed. The values for members *sysname* and *nodename* may be dynamically set using the **sysmot()** function and are set at the same value.

sysname is one of: "unix", or set to a local value.

nodename is one of: "unix", or set to a local value.

release is: "3.2" the AT&T System V source base.

version is: "0.5", "1.0", or a similar value corresponding to and identifying a particular sequential release of SYSTEM V/88 product. "0.*" connotes prerelease versions, "1.0" refers to the initial release.

machine is one of: "M68020", "M68030", or "M88100" corresponding to the type of processor on the system.

4.4.1.4 Errors

If *name* points to an invalid address, **uname**() returns -1 and sets *errno* to EFAULT.

4.5 Time

4.5.1 Get System Time

Function: **time**()

4.5.1.4 Errors

If *tloc* points to an invalid address or to an area of memory that the process does not have write permission, an exception occurs causing either a SIGSEGV or SIGBUS signal to be sent to the process.

4.5.2 Process Times

Function: **times**()

4.5.2.2 Description

There are no additional members of the *tms* structure in <**sys/times.h**>.

4.5.2.3 Returns

The arbitrary point in time used in this call is {CLK_TCK}ths of a second since system startup.

4.5.2.4 Errors

If *buffer* points to an invalid address, **times**() returns -1 and sets *errno* to EFAULT.

4.6 Environment Variables

4.6.1 Environment Access

Function: **getenv**()

4.6.1.3 Errors

There are no error conditions for **getenv**().

4.7 Terminal Identification

4.7.1 Generate Terminal Pathname

Function: **ctermid**()

4.7.1.3 Errors

There are no error conditions for **ctermid**().

4.7.2 Determine Terminal Device Name

Functions: **ttyname**(), **isatty**()

4.7.2.4 Errors

There are no error conditions for **ttyname**() or **isatty**().

4.8 Configurable System Variables

4.8.1 Get Configurable System Variables

Function: **sysconf**()

4.8.1.2 Description

Additional configurable variables from <**unistd.h**> are:

```
#define  _SC_BCS_VERSION          9  /* BCS version stamp                    */
#define  _SC_BCS_VENDOR_STAMP    10  /* Vendor stamp of system               */
#define  _SC_BCS_SYS_ID          11  /* unique machine id                    */
#define  _SC_MAXUMEMV            12  /* Max user process size 1-KB pages     */
#define  _SC_MAXUPROC            13  /* Max number of processes/user         */
#define  _SC_MAXMSGSZ            14  /* Max size of a message                */
#define  _SC_NMSGHDRS            15  /* Total number of msg headers/system   */
#define  _SC_SHMMAXSZ            16  /* Maximum size of shared segment       */
#define  _SC_SHMMINSZ            17  /* Minimum size of shared segment       */
#define  _SC_SHMSEGS             18  /* Max attached segs/process            */
#define  _SC_NMSYSSEM            19  /* Total number semaphores/system       */
#define  _SC_MAXSEMVL            20  /* Max semaphore value                  */
#define  _SC_NSEMMAP             21  /* Number of semaphore sets             */
#define  _SC_NSEMMSL             22  /* Number of semaphores/set             */
#define  _SC_NSHMMNI             23  /* Number of shared segments/system     */
#define  _SC_ITIMER_VIRT         24  /* System supports virtual timer        */
#define  _SC_ITIMER_PROF         25  /* System supports profiling timer      */
#define  _SC_TIMER_GRAN          26  /* Granularity of timers in usec        */
#define  _SC_PHYSMEM             27  /* Total physical memory/system (kb)    */
#define  _SC_AVAILMEM            28  /* Total physmem avail to user (kb)     */
#define  _SC_NICE                29  /* nice prioritization is supported     */
#define  _SC_MEMCTL_UNIT         30  /* bytes in a memory unit               */
                                     /* in memctl system call                */
#define  _SC_SHMLBA              31  /* Memory address rounding used by      */
                                     /* shmsys in bytes                      */
#define  _SC_SVSTREAMS           32  /* System V streams are supported       */
#define  _SC_CPUID               33  /* return Processor Identification
                                        Register                             */
```

5. Files and Directories

5.1 Directories

5.1.1 Format of Directory Entries

SYSTEM V/88, RFS, and NFS file systems are supported. For SYSTEM V/88 file systems, a directory is a file that contains one entry for each file contained in the directory. Directory entries are defined by the structure *direct* in <**sys/fs/s5dir.h**>.

```
#define  DIRSIZ    14
 struct direct {
        ino_t d_ino;
        char  d_name[DIRSIZ];
 };
```

The *dirent* structure in <**sys/dirent.h**> has three elements in addition to *d_name*:

```
struct dirent    /* data from readdir() */
    {
    ino_t          d_ino;              /* inode number of entry          */
    off_t          d_off;              /* offset of disk directory entry */
    unsigned short d_reclen;           /* length of this record          */
    char           d_name[MAXNAMLEN + 1]; /* name of file                */
       };
```

5.1.2 Directory Operations

Functions: **opendir**(), **readdir**(), **rewinddir**(), **closedir**()

5.1.2.2 Description

The type DIR, defined in <**dirent.h**>, uses a file descriptor. This limits the total number of open files and directories to {OPEN_MAX}. The **closedir**() function closes the file descriptor associated with the directory stream.

```
typedef struct
        {
        int     dd_fd;      /* file descriptor              */
        int     dd_loc;     /* offset in block              */
        int     dd_size;    /* amount of valid data         */
        char   *dd_buf;     /* directory block              */
        }       DIR;        /* stream data from opendir()  */
```

5.2 Working Directory

5.2.1 Change Current Working Directory

5.2.2 Working Directory Pathname

5.3 General File Creation

5.3.1 Open a File

Function: **open**()

5.3.1.2 Description

The additional flags for *oflag* in <**fcntl.h**>:

```
O_ACCMOD    Mask used to get O_RDONLY, O_WRONLY, and O_RDWR flags
O_NDELAY    SVID defined non-blocking I/O
O_SYNC      Synchronous write option
```

If bits in *mode* other than file permissions are used, the permissions on the file are undefined.

If **open** is called with O_EXCL, O_CREAT must also be present; otherwise O_EXCL is ignored.

If **open** is called with O_TRUNC and the file is not of type regular, directory, or of type FIFO, O_TRUNC has no effect.

5.3.2 Create a New File or Rewrite an Existing One

5.3.3 Set File Creation Mask

Function: **umask**()

5.3.3.2 Description

The bits other than file permission bits in *cmask* are ignored.

5.3.4 Link to a File

Function: **link**()

5.3.4.2 Description

The **link**() function supports linking of files across file systems and, when the process has *appropriate privileges*, on directories. Permission to access the existing file is not required by this implementation.

5.3.4.4 Errors

5.4 Special File Creation

5.4.1 Make a Directory

Function: **mkdir**()

5.4.1.2 Description

The bits other than file permission bits in *mode* are ignored.

5.4.2 Make a FIFO Special File

Function: **mkfifo**()

5.4.2.2 Description

If bits in *mode* other than file permissions are used, the results are undefined.

5.5 File Removal

5.5.1 Remove Directory Entries

Function: **unlink**()

5.5.1.2 Description

The implementation supports using **unlink**() on directories.

5.5.1.4 Errors

The directory named by the *path* argument can be unlinked when it is being used by the system or another process; it is not an [EBUSY] error.

The implementation supports using **unlink**() on directories, so [EPERM] error is not used for this purpose.

5.5.2 Remove a Directory

Function: **rmdir**()

5.5.2.2 Description

If an attempt is made to remove the root directory, **rmdir**() returns -1 and sets *errno* to EBUSY.

If a process attempts to remove its current working directory, **rmdir**() returns -1 and sets *errno* to EINVAL.

5.5.3 Rename a File

Function: **rename**()

5.5.3.4 Errors

5.6 File Characteristics

5.6.1 File Characteristics: Header and Data Structure

The *stat* structure in <**sys/stat.h**> is:

```
struct   stat {
            dev_t       st_dev;
            ino_t       st_ino;
            mode_t      st_mode;
            nlink_t     st_nlink;
            uid_t       st_uid;
            gid_t       st_gid;
            dev_t       st_rdev;
            off_t       st_size;
            time_t      st_atime;
            time_t      st_ausec;     /* atime extra usecs (BCS) */
            time_t      st_mtime;
            time_t      st_musec;     /* mtime extra usecs (BCS) */
            time_t      st_ctime;
            time_t      st_cusec;     /* ctime extra usecs (BCS) */
            char        st_pad[456];  /* BCS */
};
```

st_rdev is defined only for block or character devices. For these devices, *st_rdev* specifies the device id. *st_ausec*, *st_musec*, and *st_cusec* are added for compliance with BCS and are used to store microsecond resolution information for *st_atime*, *st_mtime*, and *st_ctime* respectively. *st_pad* is padding used to fulfill structure size of 512 bytes in BCS.

5.6.1.1 <sys/stat.h> File Types

5.6.1.2 <sys/stat.h> File Modes

No other bits are ORed into S_IRWXU, S_IRWXG, or S_IRWXO.

5.6.1.3 <sys/stat.h> Time Entries

mknod() changes the values of *st_atime*, *st_mtime*, and *st_ctime*.

5.6.2 Get File Status

Functions: **stat**(), **fstat**()

5.6.2.2 Description

No additional or alternate file access control mechanisms are implemented.

5.6.2.4 Errors

If *buf* points to an invalid address, **fstat**() returns -1 and sets *errno* to EFAULT.
If *buf* points to an invalid address, **stat**() returns -1 and sets *errno* to EFAULT.

5.6.3 File Accessibility

Functions: **access**()

5.6.3.2 Description

A user with effective user ID of zero is always granted execute permission even though (1) execute permission is meaningful only for directories and regular files, and (2) **exec** requires that at least one execute mode bit is set for regular files to be executable.

5.6.4 Change File Modes

Function: **chmod**()

5.6.4.2 Description

S_ISUID and S_ISGID bits are ignored if the file owner is the superuser, and the file system is an NFS remotely mounted file system.

chmod() of an open file has no effect on the file descriptor.

5.6.5 Change Owner and Group of a File

Function: **chown**()

5.6.5.2 Description

If a process with effective user ID of zero performs a **chown**(), the setuid and setgid bits are not changed.

5.6.6 Set File Access and Modification Times

5.7 Configurable Pathname Variables

5.7.1 Get Configurable Pathname Variables

Function: **pathconf**(), **fpathconf**()

5.7.1.2 Description

Additional Configurable Pathname Variables from **<unistd.h>**:

```
#define  _PC_BLKSIZE          10
#define  _PC_CHOWN_SUP_GRP    11
#define  _PC_DIR_DOTS         12
#define  _PC_GROUP_PARENT     13
#define  _PC_UTIME_OWNER      14
```

_PC_CHOWN_SUP_GRP, _PC_DIR_DOTS, _PC_GROUP_PARENT, and _PC_UTIME_OWNER are used only for PCTS draft 12 testing and have no function.

6. Input and Output Primitives

6.1 Pipes

6.1.1 Create an Inter-Process Channel

6.2 File Descriptor Manipulation

6.2.1 Duplicate an Open File Descriptor

6.3 File Descriptor Deassignment

6.3.1 Close a File

6.4 Input and Output

6.4.1 Read from a File

Function: **read**()

6.4.1.2 Description

If **read**() is interrupted by a signal after successfully reading some data, it returns the number of bytes read.

After end-of-file is reached, subsequent **read**() requests on *fildes* returns zero. This also applies to special device files.

6.4.1.4 Errors

[EIO] The implementation supports job control, the process is in a background process group and is attempting to read from its controlling terminal, and either the process is ignoring or blocking the SIGTTIN signal or the process group of the process is orphaned. This error may also be generated as a result of actual I/O errors, such as device failure.

6.4.2 Write to a File

Function: **write**()

6.4.2.2 Description

If *nbyte* is zero and the file is a device, the command will be passed to the device; the effect is device dependent.

If **write**() is interrupted by a signal after successfully writing some data, it returns the number of bytes written.

If *nbyte* is greater than INT_MAX, **write**() returns -1 and sets *errno* to EINVAL.

If the O_NONBLOCK flag is set, write requests to a pipe for {PIPE_BUF} or fewer bytes shall either succeed completely and returns nbyte, or returns -1 and sets errno to [EAGAIN], write requests for greater than {PIPE_BUF} bytes shall transfer what it can and return the number of bytes written.

6.4.2.4 Errors

EFBIG will never occur if the default value of maximum file size is used. The default maximum file size in terms of bytes is greater than ULONG_MAX; thus ULONG_MAX will be exceeded before the maximum file size is reached. The maximum file size is configurable for a process using **ulimit**(2).

If *errno* has the value EINTR following a **write**(), no data was returned.

[EIO] The implementation supports job control, the process is in a background process group and is attempting to write to its controlling terminal, TOSTOP is set, the process is neither ignoring nor blocking SIGTTOU signals, and the process group of the process is orphaned. This error may also be generated as a result of actual I/O errors, such as device failure.

6.5 Control Operations on Files

6.5.1 Data Definitions for File Control Operations

Additional File Control flags from <**sys/fcntl.h**>:

```
#define  F_CHKFL       8  /* Check legality of file flag changes      */
#define  F_ALLOCSP    10  /* reserved                                 */
#define  F_FREESP     11  /* reserved                                 */
#define  O_NDELAY     04  /* Non-blocking I/O, SVID semantics         */
#define  O_SYNC      020  /* synchronous write option                 */
```

6.5.2 File Control

Function: **fcntl()**

6.5.2.2 Description

If status bits, other than those defined, are set when **fcntl()** is called with F_SETFL as the value for *cmd*, they are ignored.

Advisory record locking is supported for regular files only.

The member *l_sysid* is in addition to members described in POSIX. The *l_sysid* field represents a system identifier that may be used when locking files on remote systems; it is defined in SVR3.

The *flock* structure defined in <**fcntl.h**>:

```
struct flock {
          short     l_type;
          short     l_whence;
          off_t     l_start;
          off_t     l_len;
          pid_t     l_pid;
          short     l_sysid;
          short     l_pad;
  };
```

If *l_len* is negative when attempting to lock, the lock succeeds. However, this is not recommended because the checks for existing locks will not find conflicts if there are locks that were specified in this way.

6.5.3 Reposition Read/Write File Offset

Function: **lseek**()

6.5.3.2 Description

This only works fully on disks. The cartridge tape supports forward seeking on reads only.

7. Device- and Class-Specific Functions

7.1 General Terminal Interface

The terminal interface supports network connections and asynchronous serial communication ports. Synchronous ports are not supported.

7.1.1 Interface Characteristics

7.1.1.1 Opening a Terminal Device File

7.1.1.2 Process Groups

7.1.1.3 The Controlling Terminal

If a session leader has no controlling terminal, and opens a terminal device file that is not already associated with a session without using the O_NOCTTY option, the terminal becomes the controlling terminal of the session leader.

Accesses to a terminal by a process in a session after the controlling process terminates and the controlling terminal is disassociated from the session is denied.

7.1.1.4 Terminal Access Control

7.1.1.5 Input Processing and Reading Data

When the input queue limit, {MAX_INPUT}, is exceeded, the input queue is flushed.

7.1.1.6 Canonical Mode Input Processing

When the input line limit, {MAX_CANON}, is exceeded, the additional characters are discarded.

7.1.1.7 Non-Canonical Mode Input Processing

MIN is stored in an unsigned character, therefore, it cannot be greater than {MAX_INPUT} 255.

7.1.1.7.1 Case A: MIN > 0, TIME > 0

7.1.1.7.2 Case B: MIN > 0, TIME = 0

7.1.1.7.3 Case C: MIN = 0, TIME > 0

7.1.1.7.4 Case D: MIN = 0, TIME = 0

7.1.1.8 Writing Data and Output Processing

There is no buffering mechanism.

7.1.1.9 Special Characters

The START and STOP characters are changeable.

There are no multi-byte special character sequences.

There are two additional single-byte special characters:

`SWTCH`	`Change current layer on an sxt device (`*shl*`)`
`VEOL2`	`Same as VEOL`

7.1.1.10 Modem Disconnect

7.1.1.11 Closing a Terminal Device File

7.1.2 Settable Parameters

7.1.2.1 *termios* Structure

There is one additional member, *c_line*, in the *termios* structure in <**termios.h**>; *c_line* specifies the line discipline number:

```
#define  NCCS     19   /* Size of the c_cc field in the struct below */
typedef  unsigned long    tcflag_t;
typedef  unsigned char    cc_t;

struct termios {
    tcflag_t     c_iflag; /* input modes              */
    tcflag_t     c_oflag; /* output modes             */
    tcflag_t     c_cflag; /* control modes            */
    tcflag_t     c_lflag; /* line discipline modes    */
    char         c_line;  /* line discipline          */
    cc_t         c_cc[NCCS]; /* control chars         */
};
```

The total size of the *termios* structure is 36 bytes.

7.1.2.2 Input Modes

In the context of terminal network connections, the break condition does not occur. Synchronous terminal connectons are not supported.

START is transmitted if the input queue is nearly empty. STOP is transmitted when the input queue is nearly full.

The initial input control value is all bits clear.

7.1.2.3 Output Modes

If OPOST is set, output characters are postprocessed as indicated by the remaining flags.

Additional flags supported for *c_oflag* are:

```
OLCUC       Map lower case to upper case on output
ONLCR       Map NL to CR-NL on output
OCRNL       Map CR to NL on output
ONOCR       No CR output at column zero
ONLRET      NL performs CR function
OFILL       Use fill characters for delay
OFDEL       Fill is DEL, else NNUL
NLDLY       Select newline delays:
  NLO
  NL1
CRDLY       Select carriage-return delays:
  CRO
  CR1
  CR2
  CR3
TABDLY      Select horizontal-tab delays:
  TABO
  TAB1
  TAB2
  TAB3
BSDLY       Select backspace delays:
  BSO
  BS1
VTDLY       Select vertical-tab delay:
  VTO
  VT1
FFDLY       Select form feed delays:
  FFO
  FF1
```

The initial output control value is all bits clear.

7.1.2.4 Control Modes

The initial hardware control values for local ttys are B9600, CS8, CREAD and HUPCL. The initial hardware control values for remote ttys (psuedo ttys) are B38400, CS8, CREAD and HUPCL.

7.1.2.5 Local Modes

If ECHOE and ICANON are set, and there is no character to erase, nothing is done.

If IEXTEN is set, VSWTCH and VEOL2 are enabled.

The initial local control value is all bits clear.

7.1.2.6 Special Control Characters

The number of elements in the *c_cc* array is the value of NCCS. NCCS is currently defined in **<termios.h>** as 19.

The initial values of the control characters are:

```
VINTR           0177    /* DEL          */
VQUIT           034     /* FS, cntl-|   */
VERASE          '#'
VKILL           '@'
VEO             04      /* cntl-d       */
VSTART          021     /* cntl-q       */
VSTOP           023     /* cntl-s       */
VSWTCH          0
VSUSP           0
```

7.1.2.7 Baud Rate Functions

Functions: **cfgetispeed**(), **cfgetospeed**(), **cfsetispeed**(), **cfsetospeed**()

7.1.2.7.2 Description

No errors are returned by **cfsetispeed**() and **cfsetospeed**(). **tcsetattr**() may return errors as described in Section 7.2.1.

7.1.2.7.4 Errors

There are no error conditions for these functions.

7.2 General Terminal Interface Control Functions

Job control is supported.

7.2.1 Get and Set State

7.2.2 Line Control Functions

Functions: **tcdrain()**, **tcflow()**, **tcflush()**, **tcsendbreak()**

7.2.2.2 Description

If *duration* is non-zero, *tcsendbreak* sends a break of length *duration**HZ/1000 in clock ticks, where there are HZ clock ticks per second. Network terminals ignore break requests.

7.2.3 Get Foreground Process Group ID

Function: **tcgetpgrp()**

7.2.3.2 Description

Job control is supported and the function is supported as described.

7.2.4 Set Foreground Process Group ID

Function: **tcsetpgrp()**

7.2.4.2 Description

Job control is supported and the function is supported as described.

8. Language-Specific Services for the C Programming Language

8.1 Referenced C Language Routines

All the routines documented in this section are described in reference to their common usage C definition, not the ANSI C definition.

8.1.1 Extensions to Time Functions

If the first character of the environment variable TZ is a colon (:), there is no special handling of the characters following the colon. Since the colon has no meaning, the defaults are used.

8.1.2 Extensions to setlocale() Function

The **setlocale**() function and its extensions are not implemented.

8.2 FILE-Type C Language Functions

8.2.1 Map a Stream Pointer to a File Descriptor

8.2.2 Open a Stream on a File Descriptor

Function: **fdopen**()

8.2.2.2 Description

There are no additional values for the *type* argument for **fdopen**(). The *st_atime* is marked for update if **fdopen()** is called with type "a" or "a+".

8.2.2.4 Errors

There are no errors returned from **fdopen**().

8.2.3 Interactions of Other FILE-Type C Functions

8.2.4 Operations on Files — the remove() Function

8.3 Other C Language Functions

8.3.1 Non-Local Jumps

8.3.2 Set Time Zone

8.3.2.2 Description

The default time zone when TZ is absent from the environment is Greenwich Mean Time (GMT).

9. System Databases

9.1 System Databases

If the initial working directory field in the password file, **/etc/passwd**, is null, the user's home directory is the **root** directory.

There are two additional fields in a password file entry. An encrypted password field follows the login name and a field for the users real name follows the numeric group ID. There is an optional comment field following the numeric group ID field.

A group file entry has an encrypted password field.

9.2 Database Access

9.2.1 Group Database Access

Functions: **getgrgid**(), **getgrnam**()

9.2.1.2 Description

In addition to members in Table 9-1, a group structure also has an encrypted password field.

The group structure in <**grp.h**>:

```
struct group {
            char        *gr_name;
            char        *gr_passwd;
            int         gr_gid;
            char        **gr_mem;
    };
```

9.2.1.4 Errors

No errors are detected or returned.

9.2.2 User Database Access

Functions: **getpwuid()**, **getpwnam()**

9.2.2.2 Description

In addition to members in Table 9-2, a passwd structure has an encrypted password field, a password aging field, a comment field, and a GCOS field used only when communicating with that system.

cuserid() calls **getpwnam()** to determine the user name associated with the effective user ID of the process; thus the results of a call to either **cuserid()** or **getpwnam()** may be overwritten by a subsequent call to the other routine.

The passwd structure in **<pwd.h>**:

```
struct passwd {
        char    *pw_name;
        char    *pw_passwd;
        int      pw_uid;
        int      pw_gid;
        char    *pw_age;
        char    *pw_comment;
        char    *pw_gecos;
        char    *pw_dir;
        char    *pw_shell;
};
```

9.2.2.4 Errors

No errors are detected or returned.

9.2.3 User Database Access

10. Data Interchange Format

10.1 Archive/Interchange File Format

Either **tar**(1) or **cpio**(1) may be used to create or read archives.

10.1.1 Extended tar Format

Filenames from an archive that exceed NAME_MAX are silently truncated.

If the *typeflag* field is set to CHARTYPE, BLKTYPE, or FIFOTYPE, the meaning of the *size* field is undefined.

Symbolic links are supported, therefore, typeflag '2' produces this type of file.

Special files are created with the major and minor numbers specified by *devmajor* and *devminor*.

The high performance attribute extension is not supported.

10.1.2 Extended cpio Format

10.1.2.1 Header

The values of *c_dev*, *c_ino* and *c_rdev* are the values in the corresponding fields of the data structure returned by **stat**().

Special files are created with the major and minor numbers specified by *st_rdev* for the file in the archive.

10.1.2.2 File Name

Filenames from an archive that exceed NAME_MAX are silently truncated.

10.1.2.3 File Data

10.1.2.4 Special Entries

10.1.2.5 cpio Values

10.1.3 Multiple Volumes

The user is prompted for the next file when end-of-file is encountered.

MOTOROLA INC.

Microcomputer Division
2900 South Diablo Way
Tempe, Arizona 85282
P.O. Box 2953
Phoenix, Arizona 85062

11044 PRINTED IN USA (3/90) WPC 2,500