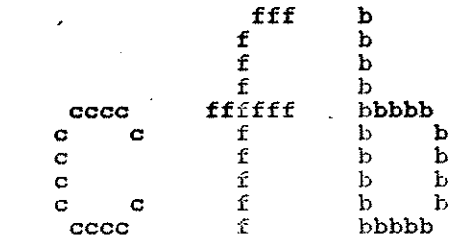

Sat Feb 9 09:01:00 1980

```
#       @(#)70.mk       2.20.1.2

INS = cpmv
FRC =

COMPOOL = /usr/include/sys
ICOMPOOL = /usr/include

HEADERS = \
        $(COMPOOL)/buf.h \
        $(COMPOOL)/bufx.h \
        $(COMPOOL)/conf.h \
        $(COMPOOL)/confx.h \
        $(COMPOOL)/crtctl.h \
        $(COMPOOL)/dir.h \
        $(COMPOOL)/dmll.h \
        $(COMPOOL)/elog.h \
        $(COMPOOL)/err.h \
        $(COMPOOL)/file.h \
        $(COMPOOL)/filex.h \
        $(COMPOOL)/ino.h \
        $(COMPOOL)/inode.h \
        $(COMPOOL)/inodex.h \
        $(COMPOOL)/iobuf.h \
        $(COMPOOL)/ipcomm.h \
        $(COMPOOL)/ipcommx.h \
        $(COMPOOL)/lfsh.h \
        $(COMPOOL)/lock.h \
        $(COMPOOL)/map.h \
        $(COMPOOL)/maus.h \
        $(COMPOOL)/mx.h \
        $(COMPOOL)/param.h \
        $(COMPOOL)/proc.h \
        $(COMPOOL)/procx.h \
        $(COMPOOL)/reg.h \
        $(COMPOOL)/rx.h \
        $(COMPOOL)/seg.h \
        $(COMPOOL)/sgtty.h \
        $(COMPOOL)/sigdef.h \
        $(COMPOOL)/sprof.h \
        $(COMPOOL)/sprofx.h \
        $(COMPOOL)/stat.h \
        $(COMPOOL)/sysmes.h \
        $(COMPOOL)/sysmesx.h \
        $(COMPOOL)/systm.h \
        $(COMPOOL)/text.h \
        $(COMPOOL)/textx.h \
        $(COMPOOL)/timeb.h \
        $(COMPOOL)/trans.h \
        $(COMPOOL)/tty.h \
        $(COMPOOL)/ttyx.h \
        $(COMPOOL)/types.h \
        $(COMPOOL)/user.h \
        $(COMPOOL)/userx.h \
```

```
        $(COMPOOL)/utsname.h \
        $(COMPOOL)/version.h \
        $(COMPOOL)/votrax.h \
        $(COMPOOL)/vtll.h \
        $(COMPOOL)/vtmn.h

LHEADERS = \
        $(COMPOOL)/filsys.h \
        $(COMPOOL)/syserr.h \
        $(COMPOOL)/ioctl.h

all install:    $(FRC) $(HEADERS) $(LHEADERS)
        @echo Headers are now up to date.

$(HEADERS):     $$(@F)
        $(INS) $?       444 src sys $@

$(COMPOOL)/syserr.h:    $$(@F)
        $(INS) $?       444 src sys $@ $(ICOMPOOL)/errno.h

$(COMPOOL)/filsys.h:    $$(@F)
        $(INS) $?       444 src sys $@ $(ICOMPOOL)/filsys.h

$(COMPOOL)/ioctl.h:     $$(@F)
        $(INS) $?       444 src sys $@ $(ICOMPOOL)/sgtty.h

clean:;

FRC:
        rm -f $(HEADERS)

.PRECIOUS:      $(HEADERS)


.h~.h:
        get -s $<
```

```
#       @(#)Makefile    2.1

70:
all::   70

.DEFAULT:
        make -f $<.mk
```

```
/*      @(#)buf.h       2.6     */

/*
 * Each buffer in the pool is usually doubly linked into 2 lists;
 * the device with which it is currently associated (always)
 * and also on a list of blocks available for allocation
 * for other use (usually).
 * The latter list is kept in last-used order, and the two
 * lists are doubly linked to make it easy to remove
 * a buffer from one list when it was found by
 * looking through the other.
 * A buffer is on the available list, and is liable
 * to be reassigned to another disk block, if and only
 * if it is not marked BUSY.  When a buffer is busy, the
 * available-list pointers can be used for other purposes.
 * Most drivers use the forward ptr as a link in their I/O
 * active queue.
 * A buffer header contains all the information required
 * to perform I/O.
 * Most of the routines which manipulate these things
 * are in bio.c.
 */
struct buf
{
        int     b_flags;                /* see defines below */
        struct  buf *b_forw;            /* headed by devtab of b_dev */
        struct  buf *b_back;            /*  "  */
        struct  buf *av_forw;           /* position on free list, */
        struct  buf *av_back;           /*     if not BUSY*/
        dev_t   b_dev;                  /* major+minor device name */
        unsigned b_bcount;              /* transfer count */
        paddr_t b_paddr;                /* physical address */
        daddr_t b_blkno;                /* block # on device */
        char    b_error;                /* returned after I/O */
        char    b_pri;                  /* priority of this request */
        unsigned int    b_resid;        /* bytes not transferred after error */
};

#define paddr(X)        X->b_paddr
#define getblk(dev,blkno)       aagetblk(dev,blkno,0)
#define agetblk(dev,blkno)      aagetblk(dev,blkno,1)
#define bread(dev,blkno)        aabread(dev,blkno,0)
#define abread(dev,blkno)       aabread(dev,blkno,1)

/*
 * These flags are kept in b_flags.
 */
#define B_WRITE 0       /* non-read pseudo-flag */
#define B_READ  01      /* read when I/O occurs */
#define B_DONE  02      /* transaction finished */
#define B_ERROR 04      /* transaction aborted */
#define B_BUSY  010     /* not on av_forw/back list */
#define B_PHYS  020     /* Physical IO potentially using UNIBUS map */
#define B_MAP   040     /* This block has the UNIBUS map allocated */
#define B_WANTED 0100   /* issue wakeup when BUSY goes off */
```

```
#define B_AGE    0200     /* delayed write for correct aging */
#define B_ASYNC  0400     /* don't wait for I/O completion */
#define B_DELWRI 01000    /* don't write till block leaves available list */
#define B_HEAD   02000    /* physio header flag */
#define B_STALE  04000
```

```
/*      @(#)bufx.h      2.4     */

/*
 * Allocation of buffer headers
 */

struct  buf     buf[NBUF+NXBUF];

/*
 * Head of the available list for buffers
 */

struct  buf     bfreelist;

/*
 * Allocation of the physio header pool
 */

struct  buf     heads[NHEAD];

/*
 * Head of the available list for headers
 * Allocation of a flag word for header management
 */

struct  buf     *hfreelist;
int             h_flags;

/*
 * Base address of external buffers
 */
paddr_t bufbase;
```

```
/*      @(#)conf.h      2.6     */

/*
 * Used to dissect integer device code
 * into major (driver designation) and
 * minor (driver parameter) parts.
 */
struct
{
        char    d_minor;
        char    d_major;
};

/*
 * Declaration of block device
 * switch. Each entry (row) is
 * the only link between the
 * main unix code and the driver.
 * The initialization of the
 * device switches is in the
 * file conf.c.
 */
struct  bdevsw
{
        int     (*d_open)();
        int     (*d_close)();
        int     (*d_strategy)();
        struct iobuf *d_tab;
};

/*
 * Character device switch.
 */
struct  cdevsw
{
        int     (*d_open)();
        int     (*d_close)();
        int     (*d_read)();
        int     (*d_write)();
        int     (*d_ioctl)();
        int     (*d_mctl)();
        struct tty *d_ttys;
};
/*
 * Declaration of line discipline
 * switch.  Each line is a link
 * between the drivers and a particular
 * type of line discipline.
 * Initialization is in conf.c
 */

struct  linesw
{
        int     (*l_read)();
        int     (*l_rcvd)();
```

```
        int     (*l_write)();
        int     (*l_xmtd)();
        int     (*l_ioctl)();
        int     (*l_dst)();
        int     (*l_open)();
        int     (*l_close)();
        int     (*l_xdma)();
} ;

/*
 * Declaration of terminal type
 * switch.  Each entry is a link
 * to a particular type of terminal
 * conversion package.
 * Initialization is in conf.c
 */

struct termsw {
        int     (*t_input)();
        int     (*t_output)();
        int     (*t_ioctl)();
} ;
```

```
/*      @(#)confx.h     2.6     */

/*
 * Nblkdev is the number of entries
 * (rows) in the block switch. It is
 * initialize in conf.70.c
 * Used in bounds checking on major
 * device numbers.
 */
int     nblkdev;

/*
 * Number of character switch entries.
 * Initialized in conf.70.c
 */
int     nchrdev;

/*
 * Nldisc is the number of entries
 * (rows) in the line discipline
 * switch.
 * Initialize in conf.70.c.
 * mc line discipline for mx is not included in the count.
 */
int     nldisc;

/*
 * Nttype is the number of entries
 * in the terminal type switch.
 * Initialized in conf.70.c
 */
int     nttype;

#ifdef MAKEMAUS
int     mauscore;               /* Start of MAUS segments */
int     mausend;                /* Last core address of MAUS regions */
int     nmausent;               /* Number of entries in mausmap */
#endif

#ifdef  NXCLIST
int     xcfreelist;
int     xclistbase;
#endif

/*
 * Allocate array names for various switch tables.
 */

struct  bdevsw  bdevsw[];

struct  cdevsw  cdevsw[];

struct  linesw  linesw[];

struct  termsw  termsw[];
```

```
/*      @(#)dir.h       2.1     */
#ifndef DIRSIZ
#define DIRSIZ                  14
#endif

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

```
/*      @(#)dm11.h      2.3     */

/*
 * DM11 control and status register bit definitions.
 */

#define DMBUSY  020     /* DM11 Busy (i.e. scanning) */
#define SCENABL 040     /* DM11 scan enable */
#define CLSCAN  04000   /* DM11 Clear scan */
#define SRTRANS 010000  /* DM11 Secondary receive transition */
#define CSTRANS 020000  /* DM11 clear to send transition */
#define CTRANS  040000  /* DM11 carrier transition */
#define RGTRANS 0100000 /* DM11 Ring transition */

/*
 * DM11 line status register bit definitions.
 */

#define LENABLE 01      /* DM11 line enable */
#define CDLEAD  02      /* DM11 data terminal ready */
#define RQSEND  04      /* DM11 request to send */
#define SUPTD   010     /* DM11 Secondary transmit */
#define SUPRD   020     /* DM11 Secondary receive */
#define CLSEND  040     /* DM11 clear to send status */
```

```
/*      @(#)elog.h      2.4     */

/*
 * Every error record has a header as follows.
 */

struct errhdr {
        int     e_type;         /* record type */
        int     e_len;          /* bytes in record (with header) */
        time_t  e_time;         /* time of day */
};

/*
 * Error record types
 */

#define E_GOTS  010             /* Start for UNIX/TS */
#define E_GORT  011             /* Start for UNIX/RT */
#define E_STOP  012             /* Stop */
#define E_TCHG  013             /* Time change */
#define E_CCHG  014             /* Configuration change */
#define E_GOCB  015             /* Start for CB-UNIX */
#define E_BLK   020             /* Block device error */
#define E_STRAY 030             /* Stray interrupt */
#define E_PRTY  031             /* Memory parity */
#define E_OVFL  040             /* Software table overflow */
#define E_PRDV  041             /* Soft filesystem error-call to prdev() */
#define E_POWER 050             /* Power-fail restart */

/*
 * Error logging startup record. One of these is
 * sent to the logging daemon when logging is
 * first activated.
 */

#define MMR3    ((physadr)0172516)
#define SYSSIZE ((physadr)0177760)

struct estart {
        struct errhdr   e_hdr;          /* record header */
        int     e_cpu;          /* cpu type */
        int     e_mmr3;         /* contents of mem mgmt reg 3 */
        long    e_syssize;      /* system memory size (11/70 only) */
        int     e_bconf;        /* block device configuration */
};

/*
 * Error logging termination record that is sent to the daemon
 * when it stops error logging.
 */

struct eend {
        struct errhdr   e_hdr;          /* record header */
        int             e_werr;         /* Number of demon write errors */
};
```

```
/*
 * A time change record is sent to the daemon whenever
 * the system's time of day is changed.
 */

struct etimchg {
        struct errhdr   e_hdr;          /* record header */
        time_t  e_ntime;        /* new time */
};

/*
 * A configuration change message is sent to
 * the error logging daemon whenever a block device driver
 * is attached or detached (MERT only).
 */

struct econfchg {
        struct errhdr   e_hdr;          /* record header */
        char    e_trudev;       /* "true" major device number */
        char    e_cflag;        /* driver attached or detached */
};

#define E_ATCH  1
#define E_DTCH  0


/*
 * "True" major device numbers. These correspond
 * to standard positions in the configuration
 * table, but are used for error logging
 * purposes only.
 */

#define RK0     0
#define RP0     1
#define RF0     2
#define TM0     3
#define TC0     4
#define HP0     5
#define HT0     6
#define HS0     7
#define RL0     8

/*
 * IO statistics are kept for each physical unit of each
 * block device (within the driver). Primary purpose is
 * to establish a guesstimate of error rates during
 * error logging.
 */

struct iostat {
        long    io_ops;         /* number of read/writes */
        long    io_misc;        /* number of "other" operations */
        unsigned io_unlog;      /* number of unlogged errors */
};
```

```
/*
 * Template for the error record that is logged by block devices.
 */

struct eblock {
        struct errhdr   e_hdr;          /* record header */
        dev_t   e_dev;          /* "true" major + minor dev number */
        physadr e_regloc;       /* controller address */
        int     e_bacty;        /* other block I/O activity */
        struct iostat   e_stats;        /* unit I/O statistics */
        int     e_bflags;       /* read/write, error, etc */
        int     e_cyloff;       /* logical dev start cyl */
        daddr_t e_bnum;         /* logical block number */
        unsigned e_bytes;       /* number of bytes to transfer */
        long    e_memadd;       /* buffer memory address */
        unsigned e_rtry;        /* number of retries */
        int     e_nreg;         /* number of device registers */
};

/*
 * Flags (selected subset of flags in buffer header)
 */

#define E_WRITE 0
#define E_READ  1
#define E_NOIO  02
#define E_PHYS  04
#define E_MAP   010
#define E_ERROR 020

/*
 * Template for the stray interrupt record that is logged
 * every time an unexpected interrupt occurs.
 */

struct estray {
        struct errhdr   e_hdr;          /* record header */
        physadr e_saddr;        /* stray loc or device addr */
        int     e_sbacty;       /* active block devices */
};

/*
 * Memory parity error record that is logged whenever one
 * of those things occurs (11/70s only).
 */

struct eparity {
        struct errhdr   e_hdr;          /* record header */
        int     e_parreg[4];    /* memory subsystem registers */
};

/*
 * Filesystem soft errors
 */
```

```
struct eprdev {
        struct errhdr   e_hdr;  /* record header */
        int             e_missed;
        dev_t           e_fsdev; /* device containing filesystem in error */
        int             e_fserr; /* type of error */
};

/*
 *      Legal values for e_fserr
 */

#define E_FSBB 0        /* Bad block */
#define E_FSBC 1        /* Bad count */
#define E_FSNS 2        /* No space  */
#define E_FSOI 3        /* Out of inodes */

/*
 * Table overflow errors.
 */

struct eovfl {
        struct  errhdr  e_hdr;
        int             e_missed;
        int             e_tabt;
};

/*
 * Legal values for e_tabt above.
 */

#define E_FILEO         0       /* File table */
#define E_PROCO         1       /* Process table */
#define E_INODEO        2       /* Inode table */
#define E_TEXTO         3       /* Text table */

/*
 * Powerfail/restart record
 */

struct epower {
        struct  errhdr  e_hdr;
};

union eunion {
        struct estart   er_strt;
        struct eend     er_end;
        struct etimchg  er_tim;
        struct econfchg er_conf;
        struct eblock   er_blk;
        struct estray   er_stry;
        struct eparity  er_prty;
        struct eovfl    er_ovfl;
        struct eprdev   er_prdev;
        struct epower   er_power;
};
```

```
typedef union eunion    err_t;
```

```
#       @(#)emerg.mk    2.1


INS = cpmv
FRC =

COMPOOL = /usr/include/util/sys

HEADERS = \
        $(COMPOOL)/param.h

all:    $(FRC) $(HEADERS)
        @echo Headers are now up to date.

$(COMPOOL)/param.h:     param.util.h
        cpmv param.util.h       --664 src sys $@

FRC:
        rm -f $(HEADERS)

.PRECIOUS:      $(HEADERS)


.h~.h:
        get -s $<
```

```
/*      @(#)err.h       2.1     */

/*
 * structure of the err buffer area
 */
struct  err{
        int             e_nslot;                /* number of errslots */
        err_t           **e_org;                /* orgion of buffer pool */
        err_t           **e_nxt;                /* next slot to allocate */
        struct  errslot{
                int     slot[8];
        }       e_slot[NESLOT];                 /* storage area */
        struct  map     e_map[(NESLOT+3)/2];    /* free space in map */
        err_t           *e_ptrs[NESLOT];        /* pointer to logged errors*/
};

extern  struct err err;

err_t   *geteslot();
err_t   *geterec();
```

```
/*      @(#)file.h      2.4     */

/*
 * One file structure is allocated
 * for each open/creat/pipe call.
 * Main use is to hold the read/write
 * pointer associated with each open
 * file.
 */
struct  file
{
        char    f_flag;
        char    f_count;        /* reference count */
        struct inode *f_inode;  /* pointer to inode structure */
        union {
                off_t   f_offset;       /* read/write character pointer */
                struct chan *f_chan;    /* mpx channel pointer */
        } f_un;
};

/* flags */
#define FREAD   01
#define FWRITE  02
#define FPIPE   04
#define FNPIPE  010
#define FMPX    020
#define FMPY    040
#define FMP     060
```

```
/*      @(#)filex.h     2.3     */

/*
 * Allocation for the file table.
 */
struct  file file[NFILE];
```

```
/*      @(#)filsys.h    2.4     */
/*
 * Definition of the unix super block.
 * The root super block is allocated and
 * read in iinit/alloc.c. Subsequently
 * a super block is allocated and read
 * with each mount (smount/sys3.c) and
 * released with unmount (sumount/sys3.c).
 * A disk block is ripped off for storage.
 * See alloc.c for general alloc/free
 * routines for free list and I list.
 */
struct  filsys
{
        char    *s_isize;       /* size in blocks of I list */
        char    *s_fsize;       /* size in blocks of entire volume */
        int     s_nfree;        /* number of in core free blocks (0-100) */
        int     s_free[100];    /* in core free blocks */
        int     s_ninode;       /* number of in core I nodes (0-100) */
        int     s_inode[100];   /* in core free I nodes */
        char    s_flock;        /* lock during free list manipulation */
        char    s_ilock;        /* lock during I list manipulation */
        char    s_fmod;         /* super block modified flag */
        char    s_ronly;        /* mounted read-only flag */
        long    s_time;         /* current date of last update */
        int     pad[40];
        int     s_tfree;        /* Total free, for subsystem examination */
        int     s_tinode;       /* Free inodes, for subsystem examination */
        char    s_fname[6];     /* File system name */
        char    s_fpack[6];     /* File system pack name */
};
```

```
/*      @(#)ino.h       2.3     */

/*
 * The inode layout as it appears on the disk.
 * This header file in not used by the system, but by programs like
 * ncheck.
 */
struct  inode
{
        int     i_mode;
        char    i_nlink;        /* directory entries */
        char    i_uid;          /* owner */
        char    i_gid;          /* group of owner */
        char    i_size0;        /* most significant of size */
        char    *i_size1;       /* least sig */
        int     i_addr[8];      /* device addresses constituting file */
        int     i_atime[2];     /* last access time */
        int     i_mtime[2];     /* last modification time */
};

/* modes */
#define IALLOC  0100000         /* file is used */
#define IFMT    060000          /* type of file */
#define         IFDIR   040000  /* directory */
#define         IFCHR   020000  /* character special */
#define         IFBLK   060000  /* block special, 0 is regular */
#define ILARG   010000          /* large addressing algorithm */
#define ISUID   04000           /* set user id on execution */
#define ISGID   02000           /* set group id on execution */
#define ISVTX   01000           /* save text, event when not current */
#define IREAD   0400            /* read, write, execute permissions */
#define IWRITE  0200
#define IEXEC   0100
```

```
/*      @(#)inode.h     2.6     */

/*
 * The I node is the focus of all
 * file activity in unix. There is a unique
 * inode allocated for each active file,
 * each current directory, each mounted-on
 * file, text file, and the root. An inode is 'named'
 * by its dev/inumber pair. (iget/iget.c)
 * Data, from mode on, is read in
 * from permanent inode on volume.
 */

#define NADDR   8
/*
 * NINDEX is the fanout at each level of the tree and must obey:
 *      2 <= NINDEX <= 15
 */
#define NINDEX  6

struct group {
        short   g_state;
        char    g_index;
        char    g_rot;
        struct  group   *g_group;
        struct  inode   *g_inode;
        struct  file    *g_file;
        short   g_rotmask;
        short   g_datq;
        struct  chan    *g_chans[NINDEX];
};

struct  inode
{
        char    i_flag;
        char    i_count;        /* reference count */
        int     i_dev;          /* device where inode resides */
        int     i_number;       /* i number, 1-to-1 with device address */
        unsigned short i_mode;
        char    i_nlink;        /* directory entries */
        char    i_uid;          /* owner */
        char    i_gid;          /* group of owner */
        char    i_size0;        /* most significant of size */
        unsigned        i_size1;        /* least sig */
        union {
                struct  {
                        daddr_t i_addr[NADDR];  /* if normal file/directory*/
                        daddr_t i_lastr;        /* last logical block read  */
                };                              /* (for read-ahead)         */
                struct  {
                        daddr_t i_rdev;                 /* i_addr[0] */
                        struct  group   i_group;        /* mpx group file */
                };
        } i_un;
};
```

```
/* flags */
#define ILOCK   01              /* inode is locked */
#define IUPD    02              /* inode has been modified */
#define IACC    04              /* inode access time to be updated */
#define IMOUNT  010             /* inode is mounted on */
#define IWANT   020             /* some process waiting on lock */
#define ITEXT   040             /* inode is pure text prototype */
#define ICHG    IUPD            /* used in v7 */

/* modes */
#define IFMT    0170000         /* type of file */
#define         IFDIR   0140000 /* directory */
#define         IFCHR   0120000 /* character special */
#define         IFBLK   0160000 /* block special */
#define         IFREG   0100000 /* regular */
#define         IFMPC   0130000 /* multiplexed char special */
#define         IFMPB   0170000 /* multiplexed block special */
#define         IFLRG   0110000 /* large regular */
#define         IFLDR   0150000 /* large directory */
#define ISUID   04000           /* set user id on execution */
#define ISGID   02000           /* set group id on execution */
#define ISVTX   01000           /* save swapped text even after use */
#define IREAD   0400            /* read, write, execute permissions */
#define IWRITE  0200
#define IEXEC   0100
```

```
/*      @(#)inodex.h    2.4     */

/*
 * Allocation of the inode table
 */
struct  inode inode[NINODE];

struct  inode   *mpxip;         /* mpx virtual inode */
```

```
/*      @(#)iobuf.h     2.2     */

/*
 * Each block device has a iobuf, which contains private state stuff
 * and 2 list heads: the b_forw/b_back list, which is doubly linked
 * and has all the buffers currently associated with that major
 * device; and the d_actf/d_actl list, which is private to the
 * device but in fact is always used for the head and tail
 * of the I/O queue for the device.
 * Various routines in bio.c look at b_forw/b_back
 * (notice they are the same as in the buf structure)
 * but the rest is private to each device driver.
 */
struct iobuf
{
        int     b_flags;                /* see below */
        struct  buf *b_forw;            /* first buffer for this dev */
        struct  buf *b_back;            /* last buffer for this dev */
        struct  buf *b_actf;            /* head of I/O queue */
        struct  buf *b_actl;            /* tail of I/O queue */
        dev_t   b_dev;                  /* major+minor device name */
        char    b_active;               /* busy flag */
        char    b_errcnt;               /* error count (for recovery) */
        err_t   *io_erec;               /* error record */
        int     io_nreg;                /* number of registers to log on errors
*   */
        physadr io_addr;                /* csr address */
        struct  iostat  *io_stp;        /* unit I/O statistics */
        short   io_s1;                  /* space for drivers to leave things */
        short   io_s2;                  /* space for drivers to leave things */
};

#define tabinit(dv,stat)        {0,0,0,0,0,makedev(dv,0),0,0,0,0,0,stat,0,0}
#define NDEVREG (sizeof(struct device)/sizeof(int))

#define B_ONCE  01      /* flag for once only driver operations */
#define B_TAPE  02      /* this is a magtape (no bdwrite) */
#define B_TIME  04      /* for timeout use */
#define B_QHASH 010     /* If set, b_forw above points to a
                           hash table, not a buffer pointer */

#define QHSHSZ  8
#define LB2QSZ  07      /* Log to the base 2 of QHSHSZ */

typedef struct {
        int     b_flags;
        struct  buf     *b_forw;
        struct  buf     *b_back;
} qhsh_t;
```

```
/*      @(#)ioctl.h     2.3.1.1 */
/*
 * structure of arg for ioctl TIOCSETP and TIOCGETP
 */
struct  ttiocb {
        char    ioc_ispeed;
        char    ioc_ospeed;
        char    ioc_erase;
        char    ioc_kill;
        short   ioc_flags;
};

/*
 * structure for old stty and gtty system calls.
 */
struct  sgttyb  {
        char    sg_ispeed;              /* input speed */
        char    sg_ospeed;              /* output speed */
        char    sg_erase;               /* erase character */
        char    sg_kill;                /* kill character */
        short   sg_flags;               /* mode flags */
};

/*
 * tty ioctl commands
 */
#define TIOCGETD        (('t'<<8)|0)    /* get line discipline */
#define TIOCSETD        (('t'<<8)|1)    /* set line discipline */
#define TIOCHPCL        (('t'<<8)|2)    /* hangup on last close */
#define TIOCMODG        (('t'<<8)|3)
#define TIOCMODS        (('t'<<8)|4)
#define TIOCSETO        (('t'<<8)|6)    /* set other bits */
#define TIOCGETO        (('t'<<8)|7)    /* get other bits */
#define TIOCGETP        (('t'<<8)|8)    /* gtty */
#define TIOCSETP        (('t'<<8)|9)    /* stty */
#define TIOCSETN        (('t'<<8)|10)   /* stty - no flush */
#define TIOCEXCL        (('t'<<8)|13)   /* set exclude */
#define TIOCNXCL        (('t'<<8)|14)   /* clr exclude */
#define TIOCHMOD        (('t'<<8)|15)
#define TIOCTSTP        (('t'<<8)|16)   /* toggle transmit stop */
#define DIOCGETP        (('d'<<8)|8)    /* get discipline parameters */
#define DIOCSETP        (('d'<<8)|9)    /* set discipline parameters */
#define DIOCSETT        (('d'<<8)|10)   /* set terminal info */
#define DIOCGETT        (('d'<<8)|11)   /* get terminal info */
#define DIOCSETS        (('d'<<8)|12)   /* set spy mode */
#define FIOCLEX         (('f'<<8)|1)    /* set auto close */
#define FIONCLEX        (('f'<<8)|2)    /* clr autoclose */
#define FIOSPIPE        (('p'<<8)|1)    /* set pipe sleep flags */
#define FIOGPIPE        (('p'<<8)|2)    /* get pipe sleep flags */
#define VIOCGETD        (('v'<<8)|0)    /* Versatec */
#define VIOCSETD        (('v'<<8)|1)    /* Versatec */

/*
 * Define standard line discipline for TIOCSETD and TIOCGETD
 */
#define STD_LTYPE       (short)0
```

```
/*
 * Define half-duplex line discipline for TIOCSETD and TIOCGETD
 */
#define HF_LTYPE        (short)4
/*
 * Format of third argument for TIOCSETD and TIOCGETD
 */
struct sgldisc {
        short   sgl_type;
};


/*
 * Following ioctl.h commands are used within the system only.
 */
#ifdef KERNEL
#define OLDSGTTY        (('i'<<8)|1)
#define GETRFP          (('i'<<8)|2)
#define GETWFP          (('i'<<8)|3)
#endif

/*
 * Modes
 */
#define HUPCL   01              /* hangup on last close */
#define XTABS   02              /* map tabs to spaces on output */
#define LCASE   04              /* upper case only terminal */
#define ECHO    010             /* echo all received characters */
#define CRMOD   020             /* map CR->LF;echo CR or LF as CR-LF */
#define RAW     040             /* raw character input */
#define ODDP    0100            /* odd parity rcvd/xmtd */
#define EVENP   0200            /* even parity rcvd/xmtd */
#define ANYP    0300            /* any parity mask */
#define NLDELAY 001400
#define TBDELAY 002000
#define CRDELAY 030000
#define VTDELAY 040000
#define BSDELAY 0100000
#define ALLDELAY 0173400

/*
 * Delay algorithms
 */
#define CR0     0
#define CR1     010000
#define CR2     020000
#define CR3     030000
#define NL0     0
#define NL1     000400
#define NL2     001000
#define NL3     001400
#define TAB0    0
#define TAB1    002000
#define NOAL    004000
#define FF0     0
#define FF1     040000
```

```
#define BS0     0
#define BS1     0100000

/*
 * Speeds
 */
#define B0      0
#define B50     1
#define B75     2
#define B110    3
#define B134    4
#define B150    5
#define B200    6
#define B300    7
#define B600    8
#define B1200   9
#define B1800   10
#define B2400   11
#define B4800   12
#define B9600   13
#define EXTA    14
#define EXTB    15

/*
 * Character length and stop bits.
 * Character length does not include parity or stop bits.
 * Ored with ioc_ospeeed.
 */
#define SETSTOP 0200    /* set to change stop or length bits */
#define ONESTOP 0000
#define TWOSTOP 0100    /* 1.5 stop bits at 75 baud */
#define BITS5   0000
#define BITS6   0020
#define BITS7   0040
#define BITS8   0060
#define SLBITS  0160    /* Mask of stop and length bits */


/*
 * structure of arg for ioctl TIOCSETO and TIOCGETO
 */
struct ttiothcb {
        short   ioth_flags;
};

/*
 * Definition of "other" bits
 */
#define TANDEMO 01              /* enable transmission of xon/xoff */
#define HDPLX   0400            /* Half duplex line */
#define NOHUP   01000           /* not dial device flag */
#define XCLUDE  02000           /* disallow future opens */
#define NOSLEEP 04000           /* dont sleep if nothing is ready */
#define TANDEMI 040000          /* enable response to xon/xoff */
#define STDTTY  0100000         /* non-standard tty escapes and kills */
```

```
/*
 * struct of arg for ioctl FIOSPIPE and FIOGPIPE
 */
struct  pipcb   {
        char    pip_rflg;       /* read flag; 0=>nosleep */
        char    pip_wflg;       /* write flag; 0=>nosleep */
};


/*
 * structure of ioctl arg for DIOCGETT and DIOCSETT
 */
struct  termcb  {
        char    st_flgs;        /* term flags */
        char    st_termt;       /* term type */
        char    st_crow;        /* gtty only - current row */
        char    st_ccol;        /* gtty only - current col */
        char    st_vrow;        /* variable row */
        char    st_lrow;        /* last row */
};

/*
 * Terminal types
 */
#define TERM_NONE       0       /* tty */
#define TERM_TEC        1       /* TEC Scope */
#define TERM_V61        2       /* DEC VT61 */
#define TERM_V10        3       /* DEC VT100 */
#define TERM_TEX        4       /* Tektronix 4023 */
#define TERM_D40        5       /* TTY Mod 40/1 */
#define TERM_H45        6       /* Hewlitt-Packard 45 */
#define TERM_D42        7       /* TTY Mod 40/2B */

/*
 * Terminal flags
 */
#define TM_NONE         0000            /* use default flags */
#define TM_SNL          0001            /* special newline flag */
#define TM_ANL          0002            /* auto newline on column 80 */
#define TM_LCF          0004            /* last col of last row special */
#define TM_CECHO        0010            /* echo terminal cursor control */
#define TM_CINVIS       0020            /* do not send esc sequences to user */
#define TM_SET          0200            /* must be on to set/reset flags */
```

```
/*      @(#)ipcomm.h    2.5.1.2 */

/*
 * Interprocess Communication Control Structures
 */

#ifdef KERNEL
/*
 * common flags
 */

#define IP_PERM 03                      /* scope permission mask */
#define IP_ANY  0                       /* system scope */
#define IP_UID  01                      /* userid scope */
#define IP_GID  02                      /* groupid scope */
#define IP_QWANT        0100            /* entry in msg queue wanted */
#define IP_WANTED       0200            /* resource is desired */

struct ipaword
{       char    ip_flag;
        char    ip_id; };

/*
 * message control
 */

#define PMSG    5                       /* message sleep priority */
#define MSGIO   02                      /* tell iomove() this is msg */
#define MSGIN   0                       /* same as B_WRITE */
#define MSGOUT  01                      /* same as B_READ */

#define MDISAB  0
#define MENAB   1
#define MSEND   2
#define MSENDW  3
#define MRECV   4
#define MRECVW  5
#define MSTAT   6
#define MSGCTL  7

struct msghdr
{       struct msghdr   *mq_forw;
        int     mq_size;
        int     mq_sender;
        int     mq_type;
};
struct msgqhdr
{       struct msghdr   *mq_forw;       /* note same position as in msghdr */
        struct msghdr   *mq_last;
        int     *mq_procp;
        char    mq_flag;
        char    mq_cnt;
        int     mq_meslim;
};
```

```
#endif

/* commands for msgctl call here */
#define SETMQLEN 0                      /*set mes q length command*/


struct mstat {
        unsigned        ms_cnt;
        unsigned        ms_maxm;
};

struct mstruct {
        int     ms_frompid;
        int     ms_type;
};
```

```
/*      @(#)ipcommx.h   2.3     */
```

```
/*      @(#)lfsh.h      2.5     */
#define LFSINS                          /* instrumented lfs */
#define LFMAGIC 53                      /* magic number */
#define SECSIZE 512                     /* sector byte size */
#define SECSIZ2 9                       /* sector byte size (power of 2) */
#define SECOFST 1                       /* free sectors at start of file sys */
#define FDOFST 2                        /* offset of file descriptors */
#define MAFLSIZE 1                      /* size of malloc freelist in sectors *
*  /
#define MAXLFD 225                      /* max # lf descriptors */
#define MAXLFS  32767                   /* maximum # of logical files */

#define LF_OPEN 1                       /* open flag */
#define LF_BUSY 2                       /* header/descriptor busy */
#define LF_WANT 4                       /* wakeup needed */

#define L_READ  1                       /* read request */
#define L_WRIT  2                       /* write request */
#define L_CREAT 3                       /* create */
#define L_DELET 4                       /* delete lf */
#define L_SWTCH 5                       /* switch lf storage */
#define L_COPY  6                       /* copy block */
#define L_SIZE  7                       /* size of lf ? */
#define L_STAT 8                        /* get statistics */
#define L_ISTAT 9                       /* initialize statistics counts */

/*
 *      ioctl data block format.
 */
struct lfcb {
        int     lf_lfn;
        int     lf_arg1;
        int     lf_arg2;
        int     lf_arg3;
};

/*
 * Logical file system header definition structure.
 */

struct lfhead {
        int     lh_nlfs;                /* # of logical files available */
        int     lh_ncyls;               /* # of cylinders in file system */
        char    lh_magic;               /* magic number */
        char    lh_trkf;                /* tracks per cylinder */
        char    lh_secf;                /* sectors per track */
        char    lh_blkf;                /* minimum file size in sectors */
        int     lh_fres;                /* starting sector of free map */
};

/*
 * Descriptor block for a logical file.
 */

struct lfdsc {
        struct lfdsc *ld_forw;          /* forward ptr to lfn list */
```

```
        struct lfdsc *ld_back;          /* back ptr to lfn list */
        struct lfdsc *ld_avforw;        /* forward ptr to avail list */
        struct lfdsc *ld_avback;        /* back ptr to avail list */
        int     ld_lfn;                 /* lfn assoc. with this lfd */
        char    ld_flag;                /* file flags */
        char    ld_spare;               /* unused */
        char    *ld_start;              /* starting blocking factor of file */
        char    *ld_size;               /* size of file in blocking factors */
        char    *ld_secsiz;             /* size of file in sectors */
};

/*
 * Logical file system layout.
 */

struct lflayout {
        struct  lfhead  lfhd;           /* overall description */
        struct  lfdsc   lfds[MAXLFD];   /* file descriptor area */
};

struct {
        int     hiword;
        int     loword;
};

/*
 * Logical File statistics structure
*/

struct lfstat
{
        long lf_lfscal;         /* # lfs calls */
        long lf_rawrc;          /* # raw read calls */
        long lf_rawwc;          /* # raw write calls */
        long lf_rawrb;          /* # raw read blocks trnsfd */
        long lf_rawwb;          /* # raw write blocks trnsfd */
        long lf_ttmhit;         /* # trans. table memory hits */
        long lf_ttmmis;         /* # trans. table memory misses */
        long lf_brc;            /* # buffered read calls */
        long lf_bwc;            /* # buffered write calls */
};
```

```
/*      @(#)lock.h      2.2     */

/* flags for locking procs and texts
 */
#define PUNLOCK  0
#define PROCLOCK 1
#define TXTLOCK  2
#define TUNLOCK  3
```

```
/*      @(#)map.h       2.1     */

struct map {
        int     m_size;
        char    *m_addr;
};
```

```
/*      @(#)maus.h      2.5     */

#

/*
 * Common Header file for MAUS routines
 */

struct mausmap {
        int boffset;
        int bsize;
};

#ifdef MAKEMAUS

struct mausmap  mausmap[];

#endif
```

```
/*      @(#)mx.h        2.8     */

#define NGROUPS         10      /* number of mpx files permitted at one time */
#define NCHANS          20      /* number of channel structures */
#define NPORTS          30      /* number of channels to i/o ports */
#define CNTLSIZ         10
#define NLEVELS         4
#define NMSIZE          50      /* max size of mxlstn file name */

/*
 * header returned on read of mpx
 */
struct  rh {
        short   index;
        short   count;
        short   ccount;
};

/*
 * head expected on write of mpx
 */
struct  wh {
        short   index;
        short   count;
        short   ccount;
        char    *data;
};

struct  mx_args {
        char    *m_name;
        int     m_cmd;
        int     m_arg[3];
};


#ifdef KERNEL
/*
 * internal structure for channel
 */

struct chan {
        short   c_flags;
        char    c_index;
        char    c_line;
        struct  group   *c_group;
        struct  file    *c_fy;
        struct  tty     *c_ttyp;
        struct  clist   c_ctlx;
        int     c_pgrp;
        struct  tty     *c_ottyp;
        char    c_oline;
        union {
                struct  clist   datq;
        } cx;
        union {
```

```
                struct  clist   datq;
                struct  chan    *c_chan;
        } cy;
        struct  clist   c_ctly;
};

struct schan {
        short   c_flags;
        char    c_index;
        char    c_line;
        struct  group   *c_group;
        struct  file    *c_fy;
        struct  tty     *c_ttyp;
        struct  clist   c_ctlx;
        int     c_pgrp;
};

/*
 * flags
 */
#define INUSE   01
#define COPEN   02
#define XGRP    04
#define YGRP    010
#define WCLOSE  020
#define ISGRP   0100
#define BLOCK   0200
#define EOTMARK 0400
#define SIGBLK  01000
#define BLKMSG  01000
#define ENAMSG  02000
#define WFLUSH  04000
#define NMBUF   010000
#define PORT    020000
#define ALT     040000

#endif




/*
 * mpxchan command codes
 */
#define MPX     5
#define MPXN    6
#define CHAN    1
#define JOIN    2
#define EXTR    3
#define ATTACH  4
```

```
#define CONNECT 7
#define DETACH  8
#define DISCON  9
#define DEBUG   10
#define NPGRP   11
#define CSIG    12
#define PACK    13

#define NDEBUGS 30
/*
 * control channel message codes
 */
#define M_WATCH 1
#define M_CLOSE 2
#define M_EOT   3
#define M_OPEN  4
#define M_BLK   5
#define M_UBLK  6
#define DO_BLK  7
#define DO_UBLK 8
#define M_IOCTL 12


/*
 * debug codes other than mpxchan cmds
 */
#define MCCLOSE 29
#define MCOPEN  28
#define ALL     27
#define SCON    26
#define MSREAD  25
#define SDATA   24
#define MCREAD  23
#define MCWRITE 22

/*
 * ioctl commands for mpx
 */
#define MXLSTN  (('x'<<8)|1)
#define MXNBLK  (('x'<<8)|2)
```

```
/*      @(#)param.util.h        2.3.1.1 */

#define KERNEL  1

/*
 * tunable variables
 */

#define NBUF    10              /* number internal buffers */
#define NFBUF   4               /* # of bufs not for mounted file systems */
#define NXBUF   1               /* number of external buffers */
#define NHEAD   5               /* size of physio header cache */
#define NINODE  100*1           /* number of in core inodes */
#define NFILE   50*1            /* number of in core file structures */
#define NMOUNT  5               /* number of mountable file systems */
#define MAXCORE (1024*32)       /* max core overall - first # is Kw */
#define MAXMEM  (64*32)         /* max core per process - first # is Kw */
#define SSIZE   20              /* initial stack size (*64 bytes) */
#define SINCR   20              /* increment of stack (*64 bytes) */
#define NOFILE  20              /* max open files per process */
#define CANBSIZ 256             /* max size of typewriter line */
#define CMAPSIZ 50              /* size of core allocation area */
#define SMAPSIZ 50              /* size of swap allocation area */
#define NCALL   25              /* max simultaneous time callouts */
#define NPROC   50*1            /* max number of processes */
#define NTEXT   30              /* max number of pure texts */
#define HZ      60              /* Ticks/second of the clock */
#define TIMEZONE (5*60)         /* Minutes westward from Greenwich */
#define DSTFLAG 1               /* Daylight Savings Time applies here */
#define NCARGS  5120            /* # characters in exec arglist */
#define NMSAV   8               /* number of saved maus areas */
#define MSGBUFS 128             /* Characters saved from error messages */
#define NMQHDR  30              /* number message q headers */
#define MSGMEM  75              /* memory for messages (*32 bytes) */
#define MMAPSIZ 52              /* size of msg allocation area */
#define MAXMLEN 212             /* max message length in bytes */
#define MAXMSGDEF       10      /* default max number msgs per process */
#define MAXMSGL 20              /*max limit to be set by msgctl*/
#define NEVT    100             /* number of semaphores */
#define NESLOT  20              /* max number of errors kept internal */
#define NIOSTAT 2               /* number of devices to collect statistics*/
                                /* NIOSTAT must be at least as big as NHP
                                   (if the hp driver is used) plus one
                                   for each other disk driver to be loaded.
                                   (e.g. rk.c, etc.) Also, insure that
                                   DK_N in drivers such as rk.c have the
                                   correct value and iostat has been informed
                                   of these values. */

/* CONDITIONAL COMPILATION FLAGS SECTION */

#define NOSPROF                 /* System profiling */
#define NOSYSACCT               /* SYSACCT must be define for process acct. */
#define NOSPY                   /* SPY must be define for the SPY feature */
#define PWR_FAIL                /* Include power fail restart code.  Must */
```

```
                                /* also set .pf = 1 in mch.s  */
#define PTLOCK                  /* process & text locking       */
#define NOVTMON                 /* system monitor */
#define NOVTDBUG                /* enable/disable vt debug lines */
#define NODDC                   /* enable DDC/SMUX for dtp testing */
#define NOCYLHIST                       /* enable disk histogram */
#define MAKENOMAUS              /* enable shared memory */
#define NONXCLIST       60      /* enable external clist */
                                /* must also set .xclist = 1 in mch.s */

/* END CONDITIONAL FLAGS SECTION */
#ifdef NXCLIST
#define NCLIST  40              /* max total internal clist size */
#endif

#ifndef NXCLIST
#define NCLIST  20              /* max total internal clist size */
#endif

/*
 * priorities
 * probably should not be
 * altered too much
 */

#define PSWP    -100
#define PINOD   -90
#define PRIBIO  -50
#define PRIDAll -40
#define PZERO   0
#define PPIPE   1
#define PWAIT   40
#define PSLEP   90
#define PUSER   100

/*
 * signals
 * dont change
 */

#define NSIG    20
/*
 * No more than 32 signals (1-33) because they are
 * stored in bits in a (long) word.
 */

#include "sys/sigdef.h"

/*
 * fundamental variables
 * don't change too often
 */
#define USIZE   16              /* size of user block (*64) */
#define PIPSIZ  4096            /* max pipe size */
#define MEMDEV  8               /* major dev number of /dev/mem */
```

```
/*
 * fundamental constants of the implementation--
 * cannot be changed easily
 */

#define NBPW    sizeof(int)     /* number of bytes in an integer */
#define BSIZE   512             /* size of secondary block (bytes) */
#define BSLOP   0               /* If some devices need bigger buffers */
#define NINDIR  (BSIZE/sizeof(daddr_t))
#define BMASK   0777            /* BSIZE-1 */
#define BSHIFT  9               /* LOG2(BSIZE) */
#define NMASK   0177            /* NINDIR-1 */
#define NSHIFT  7               /* LOG2(NINDIR) */
#define NODEV   (dev_t)(-1)
#define ROOTINO ((ino_t)1)      /* i number of all roots */
#define SUPERB  ((daddr_t)1)    /* block number of the super block */
#define DIRSIZ  14              /* max characters per directory */
#define NULL    0

/*
 * structure to access an
 * integer in bytes
 */
struct
{
        char    lobyte;
        char    hibyte;
};

/*
 * structure to access an integer
 */
struct
{
        int     integ;
};

/*
 * structure to access a long as integers
 */
struct {
        int     hiword;
        int     loword;
};

/*
 * Certain processor registers
 */
#define PS      0177776
#define KL      0177560
#define SW      0177570

/*
 * Some macros for units conversion
 */
/* Core clicks (64 bytes) to segments and vice versa */
```

```
#define ctos(x) ((x+127)/128)

#define stoc(x) ((x)<<7)

/* Core clicks (64 bytes) to disk blocks */
#define ctod(x) ((x+7)>>3)

/* clicks to bytes */
#define ctob(x) (x<<6)

/* Core clicks (64 bytes) to k (2048 bytes) */
#define ctok(x) (x>>5)

/* Core clicks (64 bytes) to 1/10k (204.8 bytes) */
#define cto1k(x)        (((x&037)*10)>>5)

/* bytes to clicks */
#define btoc(x) (((unsigned)x+63)>>6)

/* major part of a device */
#define major(x)        (int)(((unsigned)x>>8))

/* minor part of a device */
#define minor(x)        (int)(x&0377)

/* make a device number */
#define makedev(x,y)    (dev_t)((x)<<8 | (y))

#define splx(p) PS->integ = (p)

/*
 * Typedefs
 */
#include "sys/types.h"

/*
 *      OPTIONAL CONFIGURATION DATA SECTION
 *
 *      This section contains configuration dependent
 *      defines so that separate copy of drivers are
 *      not required.  These include:
 *              io/dh.c         DHADDR  NDH11   DHSCNRAT       DHSILOLV
 *              io/dmc11.c      DMCADDR NDMC11  NDMCMB
 *              io/dz.c         DZADDR  NDZ11   DZSRATE
 *              io/hp.c         HPADDR  NHP
 *              io/hps.c        HPADDR  NHP
 *              io/kl.c         KLBASE DLBASE NKL11 NDL11
 *              io/nmpipe.c     NNAMPIPE
 */
#define NHP     2
```

```
/*      @(#)param.tu.h  2.3     */

#define KERNEL  1

/*
 * tunable variables
 */

#define NBUF    10              /* number internal buffers */
#define NFBUF   4               /* # of bufs not for mounted file systems */
#define NXBUF   1               /* number of external buffers */
#define NHEAD   5               /* size of physio header cache */
#define NINODE  100*1           /* number of in core inodes */
#define NFILE   50*1            /* number of in core file structures */
#define NMOUNT  5               /* number of mountable file systems */
#define MAXCORE (1024*32)       /* max core overall - first # is Kw */
#define MAXMEM  (64*32)         /* max core per process - first # is Kw */
#define SSIZE   20              /* initial stack size (*64 bytes) */
#define SINCR   20              /* increment of stack (*64 bytes) */
#define NOFILE  20              /* max open files per process */
#define CANBSIZ 256             /* max size of typewriter line */
#define CMAPSIZ 50              /* size of core allocation area */
#define SMAPSIZ 50              /* size of swap allocation area */
#define NCALL   25              /* max simultaneous time callouts */
#define NPROC   50*1            /* max number of processes */
#define NTEXT   30              /* max number of pure texts */
#define HZ      60              /* Ticks/second of the clock */
#define TIMEZONE (5*60)         /* Minutes westward from Greenwich */
#define DSTFLAG 1               /* Daylight Savings Time applies here */
#define NCARGS  5120            /* # characters in exec arglist */
#define NMSAV   8               /* number of saved mount areas */
#define MSGBUFS 128             /* Characters saved from error messages */
#define NMSGHDR 30              /* number message q headers */
#define MSGMEM  75              /* memory for messages (*32 bytes) */
#define MSGMAPSIZ 52            /* size of msg allocation area */
#define NSEVT   100             /* number of semaphores */
#define NESLOT  20              /* max number of errors kept internal */
#define NIOSTAT 2               /* number of devices to collect statistics*/
                                /* NIOSTAT must be at least as big as NHP
                                   (if the hp driver is used) plus one
                                   for each other disk driver to be loaded,
                                   (e.g. rk.c, etc.) Also, insure that
                                   DK_N in drivers such as rk.c have the
                                   correct value and iostat has been informed
                                   of these values. */

/* CONDITIONAL COMPILATION FLAGS SECTION */

#define NOSPROF                 /* System profiling */
#define NOSYSACCT               /* SYSACCT must be define for process acct. */
#define NOSPY                   /* SPY must be define for the SPY feature */
#define PWR_FAIL                /* Include power fail restart code.  Must */
                                /* also set .pf = 1 in mch.s  */
#define PTLOCK                  /* process & text locking       */
#define NOVTMON                 /* system monitor */
```

```
#define NOVTDBUG                /* enable/disable vt debug lines */
#define NODDC                   /* enable DDC/SMUX for dtp testing */
#define NOCYLHIST                       /* enable disk histogram */
#define MAKENOMAUS              /* enable shared memory */
#define NONXCLIST       60      /* enable external clist */
                                /* must also set .xclist = 1 in mch.s */

/* END CONDITIONAL FLAGS SECTION */
#ifdef NXCLIST
#define NCLIST  40              /* max total internal clist size */
#endif

#ifndef NXCLIST
#define NCLIST  20              /* max total internal clist size */
#endif

/*
 * priorities
 * probably should not be
 * altered too much
 */

#define PSWP    -100
#define PINOD   -90
#define PRIBIO  -50
#define PRIDA11 -40
#define PZERO   0
#define PPIPE   1
#define PWAIT   40
#define PSLEP   90
#define PUSER   100

/*
 * signals
 * dont change
 */

#define NSIG    20
/*
 * No more than 32 signals (1-33) because they are
 * stored in bits in a (long) word.
 */

#include "sys/sigdef.h"

/*
 * fundamental variables
 * don't change too often
 */
#define USIZE   16              /* size of user block (*64) */
#define PIPSIZ  4096            /* max pipe size */
#define MEMDEV  8               /* major dev number of /dev/mem */

/*
 * fundamental constants of the implementation--
 * cannot be changed easily
```

```
 */

#define NBPW    sizeof(int)     /* number of bytes in an integer */
#define BSIZE   512             /* size of secondary block (bytes) */
#define BSLOP   0               /* If some devices need bigger buffers */
#define NINDIR  (BSIZE/sizeof(daddr_t))
#define BMASK   0777            /* BSIZE-1 */
#define BSHIFT  9               /* LOG2(BSIZE) */
#define NMASK   0177            /* NINDIR-1 */
#define NSHIFT  7               /* LOG2(NINDIR) */
#define NODEV   (dev_t)(-1)
#define ROOTINO ((ino_t)1)      /* i number of all roots */
#define SUPERB  ((daddr_t)1)    /* block number of the super block */
#define DIRSIZ  14              /* max characters per directory */
#define NULL    0

/*
 * structure to access an
 * integer in bytes
 */
struct
{
        char    lobyte;
        char    hibyte;
};

/*
 * structure to access an integer
 */
struct
{
        int     integ;
};

/*
 * structure to access a long as integers
 */
struct {
        int     hiword;
        int     loword;
};

/*
 * Certain processor registers
 */
#define PS      0177776
#define KL      0177560
#define SW      0177570

/*
 * Some macros for units conversion
 */
/* Core clicks (64 bytes) to segments and vice versa */
#define ctos(x) ((x+127)/128)

#define stoc(x) ((x)<<7)
```

```
/* Core clicks (64 bytes) to disk blocks */
#define ctod(x) ((x+7)>>3)

/* clicks to bytes */
#define ctob(x) (x<<6)

/* Core clicks (64 bytes) to k (2048 bytes) */
#define ctok(x) (x>>5)

/* Core clicks (64 bytes) to 1/10k (204.8 bytes) */
#define ctolk(x)        (((x&037)*10)>>5)

/* bytes to clicks */
#define btoc(x) (((unsigned)x+63)>>6)

/* major part of a device */
#define major(x)        (int)(((unsigned)x>>8))

/* minor part of a device */
#define minor(x)        (int)(x&0377)

/* make a device number */
#define makedev(x,y)    (dev_t)((x)<<8 | (y))

#define splx(p) PS->integ = (p)

/*
 * Typedefs
 */
#include "sys/types.h"

/*
 *      OPTIONAL CONFIGURATION DATA SECTION
 *
 *      This section contains configuration dependent
 *      defines so that separate copy of drivers are
 *      not required.  These include:
 *              io/dh.c         DHADDR  NDH11   DHSCNRAT       DHSILOLV
 *              io/dmc11.c      DMCADDR NDMC11  NDMCMB
 *              io/dz.c         DZADDR  NDZ11   DZSRATE
 *              io/hp.c         HPADDR  NHP
 *              io/hps.c        HPADDR  NHP
 *              io/kl.c         KLBASE DLBASE NKL11 NDL11
 *              io/nmpipe.c     NNAMPIPE
 */
#define NHP     2
```

```
/*      @(#)param.util.h        2.3.1.1 */

#define KERNEL  1

/*
 * tunable variables
 */

#define NBUF    10              /* number internal buffers */
#define NFBUF   4               /* # of bufs not for mounted file systems */
#define NXBUF   1               /* number of external buffers */
#define NHEAD   5               /* size of physio header cache */
#define NINODE  100*1           /* number of in core inodes */
#define NFILE   50*1            /* number of in core file structures */
#define NMOUNT  5               /* number of mountable file systems */
#define MAXCORE (1024*32)       /* max core overall - first # is Kw */
#define MAXMEM  (64*32)         /* max core per process - first # is Kw */
#define SSIZE   20              /* initial stack size (*64 bytes) */
#define SINCR   20              /* increment of stack (*64 bytes) */
#define NOFILE  20              /* max open files per process */
#define CANBSIZ 256             /* max size of typewriter line */
#define CMAPSIZ 50              /* size of core allocation area */
#define SMAPSIZ 50              /* size of swap allocation area */
#define NCALL   25              /* max simultaneous time callouts */
#define NPROC   50*1            /* max number of processes */
#define NTEXT   30              /* max number of pure texts */
#define HZ      60              /* Ticks/second of the clock */
#define TIMEZONE (5*60)         /* Minutes westward from Greenwich */
#define DSTFLAG 1               /* Daylight Savings Time applies here */
#define NCARGS  5120            /* # characters in exec arglist */
#define NMSAV   8               /* number of saved mans areas */
#define MSGBUFS 128             /* Characters saved from error messages */
#define NMQHDR  30              /* number message q headers */
#define MSGMEM  75              /* memory for messages (*32 bytes) */
#define MMAPSIZ 52              /* size of msg allocation area */
#define MAXMLEN 212             /* max message length in bytes */
#define MAXMSGDEF       10      /* default max number msgs per process */
#define MAXMSGL 20              /*max limit to be set by msgctl*/
#define NEVT    100             /* number of semaphores */
#define NESLOT  20              /* max number of errors kept internal */
#define NIOSTAT 2               /* number of devices to collect statistics*/
                                /* NIOSTAT must be at least as big as NHP
                                   (if the hp driver is used) plus one
                                   for each other disk driver to be loaded.
                                   (e.g. rk.c, etc.) Also, insure that
                                   DK_N in drivers such as rk.c have the
                                   correct value and iostat has been informed
                                   of these values. */

/* CONDITIONAL COMPILATION FLAGS SECTION */

#define NOSPROF                 /* System profiling */
#define NOSYSACCT               /* SYSACCT must be define for process acct. */
#define NOSPY                   /* SPY must be define for the SPY feature */
#define PWR_FAIL                /* Include power fail restart code.  Must */
```

```
                                /* also set .pf = 1 in mch.s  */
#define PTLOCK                  /* process & text locking       */
#define NOVTMON                 /* system monitor */
#define NOVTDBUG                /* enable/disable vt debug lines */
#define NODDC                   /* enable DDC/SMUX for dtp testing */
#define NOCYLHIST                       /* enable disk histogram */
#define MAKENOMAUS              /* enable shared memory */
#define NONXCLIST       60      /* enable external clist */
                                /* must also set .xclist = 1 in mch.s */

/* END CONDITIONAL FLAGS SECTION */
#ifdef NXCLIST
#define NCLIST  40              /* max total internal clist size */
#endif

#ifndef NXCLIST
#define NCLIST  20              /* max total internal clist size */
#endif

/*
 * priorities
 * probably should not be
 * altered too much
 */

#define PSWP    -100
#define PINOD   -90
#define PRIBIO  -50
#define PRIDA11 -40
#define PZERO   0
#define PPIPE   1
#define PWAIT   40
#define PSLEP   90
#define PUSER   100

/*
 * signals
 * dont change
 */

#define NSIG    20
/*
 * No more than 32 signals (1-33) because they are
 * stored in bits in a (long) word.
 */

#include "sys/sigdef.h"

/*
 * fundamental variables
 * don't change too often
 */
#define USIZE   16              /* size of user block (*64) */
#define PIPSIZ  4096            /* max pipe size */
#define MEMDEV  8               /* major dev number of /dev/mem */
```

```
/*
 * fundamental constants of the implementation--
 * cannot be changed easily
 */

#define NBPW    sizeof(int)     /* number of bytes in an integer */
#define BSIZE   512             /* size of secondary block (bytes) */
#define BSLOP   0               /* If some devices need bigger buffers */
#define NINDIR  (BSIZE/sizeof(daddr_t))
#define BMASK   0777            /* BSIZE-1 */
#define BSHIFT  9               /* LOG2(BSIZE) */
#define NMASK   0177            /* NINDIR-1 */
#define NSHIFT  7               /* LOG2(NINDIR) */
#define NODEV   (dev_t)(-1)
#define ROOTINO ((ino_t)1)      /* i number of all roots */
#define SUPERB  ((daddr_t)1)    /* block number of the super block */
#define DIRSIZ  14              /* max characters per directory */
#define NULL    0

/*
 * structure to access an
 * integer in bytes
 */
struct
{
        char    lobyte;
        char    hibyte;
};

/*
 * structure to access an integer
 */
struct
{
        int     integ;
};

/*
 * structure to access a long as integers
 */
struct {
        int     hiword;
        int     loword;
};

/*
 * Certain processor registers
 */
#define PS      0177776
#define KL      0177560
#define SW      0177570

/*
 * Some macros for units conversion
 */
/* Core clicks (64 bytes) to segments and vice versa */
```

```
#define ctos(x) ((x+127)/128)

#define stoc(x) ((x)<<7)

/* Core clicks (64 bytes) to disk blocks */
#define ctod(x) ((x+7)>>3)

/* clicks to bytes */
#define ctob(x) (x<<6)

/* Core clicks (64 bytes) to k (2048 bytes) */
#define ctok(x) (x>>5)

/* Core clicks (64 bytes) to 1/10k (204.8 bytes) */
#define ctolk(x)        (((x&037)*10)>>5)

/* bytes to clicks */
#define btoc(x) (((unsigned)x+63)>>6)

/* major part of a device */
#define major(x)        (int)(((unsigned)x>>8))

/* minor part of a device */
#define minor(x)        (int)(x&0377)

/* make a device number */
#define makedev(x,y)    (dev_t)((x)<<8 | (y))

#define splx(p) PS->integ = (p)

/*
 * Typedefs
 */
#include "sys/types.h"

/*
 *      OPTIONAL CONFIGURATION DATA SECTION
 *
 *      This section contains configuration dependent
 *      defines so that separate copy of drivers are
 *      not required.  These include:
 *              io/dh.c         DHADDR  NDH11   DHSCNRAT       DHSILOLV
 *              io/dmc11.c      DMCADDR NDMC11  NDMCMB
 *              io/dz.c         DZADDR  NDZ11   DZSRATE
 *              io/hp.c         HPADDR  NHP
 *              io/hps.c        HPADDR  NHP
 *              io/kl.c         KLBASE DLBASE NKL11 NDL11
 *              io/nmpipe.c     NNAMPIPE
 */
#define NHP     2
```

```
/*      @(#)proc.h      2.5     */

/*
 * One structure allocated per active
 * process. It contains all data needed
 * about the process while the
 * process may be swapped out.
 * Other per process data (user.h)
 * is swapped with the process.
 */
struct  proc {
        char    p_stat;
        char    p_flag;
        char    p_pri;          /* priority, negative is high */
        char    p_time;         /* resident time for scheduling */
        char    p_cpu;          /* cpu usage for scheduling */
        char    p_nice;         /* nice for cpu usage */
        long    p_sig;          /* signals pending to this process */
        char    p_uid;          /* user id, used to direct tty signals */
        char    p_ctime;        /* length of time in current state */
        int     p_pgrp;         /* name of process group leader */
        int     p_pid;          /* unique process id */
        int     p_ppid;         /* process id of parent */
        int     p_addr;         /* address of swappable image */
        int     p_size;         /* size of swappable image (*64 bytes) */
        int     p_wchan;        /* event process is awaiting */
        int     *p_textp;       /* pointer to text structure */
        int     *p_link;        /* linked list of running processes */
        int     p_clktim;       /* time to alarm clock signal */
};

/* stat codes */
#define SSLEEP  1               /* awaiting an event */
#define SWAIT   2               /* (abandoned state) */
#define SRUN    3               /* running */
#define SIDL    4               /* intermediate state in process creation */
#define SZOMB   5               /* intermediate state in process termination */
#define SSTOP   6               /* process being traced */

/* flag codes */
#define SLOAD   01              /* in core */
#define SSYS    02              /* scheduling process */
#define SLOCK   04              /* process cannot be swapped */
#define SSWAP   010             /* process is being swapped out */
#define STRC    020             /* process is being traced */
#define SWTED   040             /* another tracing flag */
                                /* 0100 -- unused; was SSLEP */
#define SULOCK  0200            /* user setable lock in core */
                                /* research defines SULOCK as 0100 */

/*
 * Structure used to access saved times and status
 * of a dead processs, by the parent.
 * Overlays the proc structure.
 */
```

```
struct  xproc {
        char    p_stat;
        char    p_flag;
        char    p_pri;          /* priority, negative is high */
        char    p_time;         /* resident time for scheduling */
        char    p_cpu;          /* cpu usage for scheduling */
        char    p_nice;         /* nice for cpu usage */
        long    p_sig;          /* signals pending to this process */
        char    p_uid;          /* user id, used to direct tty signals */
        char    p_ctime;        /* length of time in current state */
        int     p_pgrp;         /* name of process group leader */
        int     p_pid;          /* unique process id */
        int     p_ppid;         /* process id of parent */
        int     p_addr;         /* Swap block pointer */
        int     xp_xstat;       /* Exit status for wait */
};
```

```
/*      @(#)procx.h     2.3     */

/*
 * Allocation of the proc table
 */

struct  proc proc[NPROC];
struct proc *procend;   /* highwater mark of proc table */
```

```
/*      @(#)reg.h       2.4     */

/*
 * Location of the users' stored
 * registers relative to R0.
 * Usage is u.u_ar0[XX].
 */
#define R0      (0)
#define R1      (-2)
#define R2      (-9)
#define R3      (-8)
#define R4      (-7)
#define R5      (-6)
#define R6      (-3)
#define R7      (1)
#define RPS     (2)
#define SP      R6
#define PC      R7

#define TBIT    020             /* PS trace bit */
```

```
/*      @(#)rx.h        2.1     */
/*
 *      DEFINE SYMBOLS AND STRUCTURE FOR GETTING/SETTING DENSITY
 *      ON RX02 FLOPY DRIVER.
 */

#define RIOCGETD                ('r'<<8)|0)     /* get density */
#define RIOCSETD                ('r'<<8)|1)     /* set density */

struct rx
{
        int rx_density;
};
```

```
/*      @(#)seg.h       2.5     */

/*
 * KT-11 addresses and bits.
 */

#define KISA6   0172354         /* Kernel instruction space address reg 6 */
#define KDSA    0172360         /* first kernel D-space address register */
#define UISD    0177600         /* first user I-space descriptor register */
#define UISA    0177640         /* first user I-space address register */
#define UDSA    0177660         /* first user D-space address register */
#define UDSD    0177620         /* first user D-space descrpitor register */
#define SDSA    0172260         /* first supv. D-space address register */
#define SDSD    0172220         /* first supv. D-space desc. register */
#define RO      02              /* access abilities */
#define RW      06
#define ED      010             /* extend direction */
#define TX      020             /* Software: text segment */
#define ABS     040             /* Software: absolute segment */

/*
 * structure used to address
 * a sequence of integers.
 */
struct
{
        int     r[];
};
int     *ka6;           /* 11/40 KISA6; 11/45 KDSA6 */

/*
 * address to access 11/70 UNIBUS map
 */
#define UBMAP   0170200
```

```
/*      @(#)sgtty.h     2.6     */

/*
 * stty and gtty structure layouts
 *
 * All structures are 6 bytes.
 * For each given command, doing a stty
 * sets the information into the operating
 * system. Doing a gtty retrieves it.
 */


/*
 * Command 0 --- set modes and speeds.
 *      Wait for output to drain and flush any input.
 * Command 1 -- set modes and speeds.
 *      Don't wait or flush.
 */
#define STTY_MODES      0
#define STTY_NFMODES    1
struct SGBUF {
        char    sm_ispeed;      /* Input speed */
        char    sm_ospeed;      /* Output speed, data and stop bits */
        char    sm_cmd;         /* Command = 0 or 1 */
        char    sm_fill;
        int     sm_modes;       /* See below */
};

/*
 * Modes
 */
#define NCDELAY 0000001         /* no carriage return delay */
#define XTABS   0000002         /* map tabs to spaces on output */
#define LCASE   0000004         /* upper case only terminal */
#define ECHO    0000010         /* echo all received chars */
#define CRMOD   0000020         /* map CR->LF;echo CR or LF as CR-LF*/
#define RAW     0000040         /* raw character input */
#define ODDP    0000100         /* odd parity rcvd/xmtd */
#define EVENP   0000200         /* even parity rcvd/xmtd */
#define ANYP    0000300         /* any parity mask */
#define HDPLX   0000400         /* Half duplex line */
#define NOHUP   0001000         /* don't drop DTR on last close */
#define XCLUDE  0002000         /* disallow future opens */
#define NOSLEEP 0004000         /* dont sleep if nothing is ready */
#define NTDELAY 0010000         /* no tab delay flag */
#define NLDELAY 0020000         /* no newline delay flag */
#define TANDEM  0040000         /* xon/xoff enabled */
#define STDTTY  0100000         /* non-std tty escapes and kills */

/*
 * Speeds
 */
#define B0      0
#define B50     1
#define B75     2
```

```
#define B110    3
#define B134    4
#define B150    5
#define B200    6
#define B300    7
#define B600    8
#define B1200   9
#define B1800   10
#define B2400   11
#define B4800   12
#define B9600   13
#define EXTA    14
#define EXTB    15

/*
 * Character length and stop bits.
 * Character length does not include parity or stop bits.
 * Ored with sm_ospeeed.
 */
#define SETSTOP 0200    /* set to change stop or length bits */
#define ONESTOP 0000
#define TWOSTOP 0100    /* 1.5 stop bits at 75 baud */
#define BITS5   0000
#define BITS6   0020
#define BITS7   0040
#define BITS8   0060
#define SLBITS  0160    /* Mask of stop and length bits */


/*
 * Command 2 -- set line
 * discipline of a line
 */
#define STTY_LTYPE      2

/*
 * standard line discipline
 */
#define STDLTYPE        0
struct {
        int     sl_fill;
        char    sl_cmd;         /* Command = 2 */
        char    sl_ltype;       /* Line discipline number = 0 */
        int     sl_fil2;
};

/*
 * line disciplines 1 and 2 reserved for
 * project specific line disciplines
 */
#define PRJ1LTYPE       1
#define PRJ2LTYPE       2

/*
 * transparent line discipline
 */
```

```
#define TRSLTYPE        3
struct {
        char    ts_quanta;      /* Sleep quanta */
        char    ts_fill;
        char    ts_cmd;         /* Command = 2 */
        char    ts_ltype;       /* Line discipline number = 3 */
        char    ts_brk0;        /* First break character */
        char    ts_brk1;        /* Second break character */
};

/*
 * Half Duplex line discipline
 */
#define HFLTYPE 4
struct {
        int     sl_fill;
        char    sl_cmd;         /* Command = 2 */
        char    sl_ltype;       /* Line discipline number = 4 */
        int     sl_fil2;
};

/*
 * Line disciplines 5 through 9 reserved for
 * future common line disciplines
 */
#define RSV5LTYPE       5
#define RSV6LTYPE       6
#define RSV7LTYPE       7
#define RSV8LTYPE       8
#define RSV9LTYPE       9

/*
 * Command 3 -- set terminal type
 */
#define STTY_TERM       3
struct {
        char    st_flgs;        /* terminal flags (see below) */
        char    st_fill;
        char    st_cmd;         /* Command = 3 */
        char    st_term;        /* Terminal type */
        int     st_fil2;
};

/*
 * Terminal types
 */
#define TERM_NONE       0       /* tty */
#define TERM_TEC        1       /* TEC Scope */
#define TERM_V61        2       /* DEC VT61 */
#define TERM_V10        3       /* DEC VT100 */
#define TERM_TEX        4       /* Tektronix 4023 */
#define TERM_D40        5       /* TTY Mod 40/1 */
#define TERM_H45        6       /* Hewlitt-Packard 45 */
#define TERM_D42        7       /* TTY Mod 40/2B */

/*
```

```
 * Terminal flags
 */
#define TM_NONE 0               /* use default flags */
#define TM_SNL  1               /* special newline flag */
#define TM_ANL  2               /* auto newline on column 80 */
#define TM_LCF  4               /* last col of last row special */
#define TM_CECHO        010     /* echo terminal cursor control */
#define TM_CINVIS       020     /* do not send esc sequences to user */
#define TM_SET  0200            /* must be on to set/reset flags */


/*
 * Command 4 -- set variable portion
 * of crt screen
 */
#define STTY_SCREEN     4
struct {
        char    ss_crow;        /* cursor's row */
                                /* ignored on stty */
        char    ss_fill;
        char    ss_cmd;         /* Command = 4 */
        char    ss_vrow;        /* variable row */
        int     ss_fil2;
};


/*
 * Command 0377 -- enable spy
 */
#define STTY_SPY        0377
struct {
        int     sy_fill;
        char    sy_cmd;         /* Command = 0377 */
        char    sy_scmd;        /* 0=>delete spy; 1=>initiate spy */
        int     sy_fil2;
};


/*
 * stty info for named pipes ONLY
 */
#define STTY_NPIPE      0376
struct {
        int     sp_rflg;        /* read flag; 0 => nosleep */
        char    sp_cmd;         /* Command = 0376 */
        char    sp_fill;
        int     sp_wflg;        /* write flag; 0 => nosleep */
};
```

```
/*      @(#)sigdef.h    2.3     */

#define SIGHUP  1       /* hangup */
#define SIGINT  2       /* interrupt (rubout) */
#define SIGQIT  3       /* quit (FS) */
#define SIGINS  4       /* illegal instruction */
#define SIGTRC  5       /* trace or breakpoint */
#define SIGIOT  6       /* iot */
#define SIGEMT  7       /* emt */
#define SIGFPT  8       /* floating exception */
#define SIGKIL  9       /* kill, uncatchable termination */
#define SIGBUS  10      /* bus error */
#define SIGSEG  11      /* segmentation violation */
#define SIGSYS  12      /* bad system call */
#define SIGPIPE 13      /* end of pipe */
#define SIGCLK  14      /* alarm clock */
#define SIGTRM  15      /* Catchable termination */
#define SIGCLD  18      /* death of a child */
#define SIGPWR  19      /* power failure */
```

```
/*      @(#)sprof.h     2.8     */

/*
 *
 * Used by system profiling routines( sprofil,sincupc and sprof )
 *
 *
*/

#ifdef KERNEL
struct  pgreg {
        char *par;
        char *pdr;
};

struct sysprof {
        struct  SPCNT   *base;
        caddr_t lowpc;          /* low pc word for i option */
        unsigned int numcnts;   /* number of counters in union array */
        unsigned int intsize;   /* size of i intervals or 0 for r opt */
        int pid;
        struct pgreg  newpg;
        struct pgreg  oldpg;
};

#endif
struct  NHIT {
        caddr_t nloc;
        spcnt_t nhits;
};

struct  SPCNT {
        long    b_urhits;
        long    b_syhits;
        long    b_idhits;
        union   {
                /*
                 *      "allocate" maximum possible size of counter buffers
                 *      (they must fit entirely into one page)
                 */
                struct  NHIT    ropt[(8192 - 3*sizeof(long)) / sizeof(struct NH
*  IT)];
                spcnt_t         iopt[(8192 - 3*sizeof(long))/sizeof(spcnt_t)];
        }       u_ct;
};

#ifdef  IPROFCLK

/* independent profile clock kw11-k (A clock) */

#define KW11K   (struct kw11ka *)0170404

struct  kw11ka {
        int     kw11ks;
        int     kw11kb;
```

```
};
#else
#ifdef  IPROFCLB

/* independent profile clock TCU-100 (battery clock) */

#define TCU100  (int *)0160774
#define TCURATE -48
        /*      rate of -33 should be 62.06/sec, is 120/sec for our clock
                        -45             45.6            70.6
                        -64             31              42.6
                        -48             42.6            64
                our clock may be dumb, but at least it's consistent
         */

#endif
#endif
```

```
/*      @(#)sprofx.h    2.3     */

/*
 * reserve memory for system profiling
*/

extern struct sysprof sysprof;
```

```
/*      @(#)stat.h      2.1     */
struct  stat
{
        dev_t   st_dev;
        ino_t   st_ino;
        int             st_mode;
        int             st_nlink;
        int             st_uid;
        int             st_gid;
        dev_t   st_rdev;
        off_t   st_size;
        time_t  st_atime;
        time_t  st_mtime;
        time_t  st_ctime;
};

#define S_IFMT  0170000         /* type of file */
#define         S_IFDIR 0040000 /* directory */
#define         S_IFCHR 0020000 /* character special */
#define         S_IFBLK 0060000 /* block special */
#define         S_IFREG 0100000 /* regular */
#define         S_IFMPC 0030000 /* multiplexed char special */
#define         S_IFMPB 0070000 /* multiplexed block special */
#define S_ISUID 0004000         /* set user id on execution */
#define S_ISGID 0002000         /* set group id on execution */
#define S_ISVTX 0001000         /* save swapped text even after use */
#define S_IREAD 0000400         /* read permission, owner */
#define S_IWRITE        0000200         /* write permission, owner */
#define S_IEXEC 0000100         /* execute/search permission, owner */
```

```
/*      @(#)syserr.h    2.5     */

/*
 * error codes returned from
 * various system calls.
 * found in external location
 * "errno"
 */

#define EPERM   1
#define ENOENT  2
#define ESRCH   3
#define EINTR   4
#define EIO     5
#define ENXIO   6
#define E2BIG   7
#define ENOEXEC 8
#define EBADF   9
#define ECHILD  10
#define EAGAIN  11
#define ENOMEM  12
#define EACCES  13
#ifndef KERNEL
#define EFAULT  14
#endif
#define ENOTBLK 15
#define EBUSY   16
#define EEXIST  17
#define EXDEV   18
#define ENODEV  19
#define ENOTDIR 20
#define EISDIR  21
#define EINVAL  22
#define ENFILE  23
#define EMFILE  24
#define ENOTTY  25
#define ETXTBSY 26
#define EFBIG   27
#define ENOSPC  28
#define ESPIPE  29
#define EROFS   30
#define EMLINK  31
#define EPIPE   32
#define ETABLE  33              /* no table entries left */
#define EFUNC   34              /* invalid operation requested */
#define ENOMSG  35              /* no message on queue */
#define ENOALOC 36              /* resource not allocated or improper use */
#define ELOCK   37              /* improper lock/unlock sequence */

/* math software */
/* The following should not overlap with the above like they do now */
#define EDOM    33
#define ERANGE  34
```

```
/*      @(#)sysmes.h    2.5     */

#
/* sysmes.h -- Version 02 -- Operating system measurement definitions
*/

struct M_SYSMEAS {
        unsigned m_dqlen;       /*disk queue length*/
        unsigned m_dqocc;       /*time disk queue occupied*/
        unsigned m_sqlen;       /*swap queue length*/
        unsigned m_sqocc;       /*time swap queue occupied*/
        unsigned m_rqlen;       /*run queue length*/
        unsigned m_rqocc;       /*time run queue occupied*/
        long m_termin;  /*terminal input character count*/
        long m_termop;  /*terminal output character count*/
        long m_swap;    /*swap count*/
        unsigned m_fork;        /*fork count*/
        unsigned m_exec;        /*exec count*/
        char m_povfl;   /*process table overflow count*/
        char m_fovfl;   /*file table overflow count*/
        char m_iovfl;   /*inode table overflow count*/
        char m_tovfl;   /*text table overflow count*/
        long m_dread;   /*block read count*/
        long m_dwrit;   /*block write count*/
        long m_swtch;   /*process switches*/
        long m_noblk;   /*no free buffer available*/
        long m_fndblk;  /*desired block found in core*/
        long m_gbcnt;   /*block has been requested*/
        long m_physio;  /*error free calls to physio*/
        long m_nphys;   /*blockages in physio (like noblk)*/
        long m_agbcnt;  /*addressable block has been requested*/
        long m_anoblk;  /*no free addressable buffer available*/
};
```

```
/*      @(#)sysmesx.h   2.3     */

/*sysmesx.h -- defines space for system measurement structure
*/

struct M_SYSMEAS meas;
```

```
/*      @(#)systm.h     2.7     */

/*
 * Random set of variables
 * used by more than one
 * routine.
 */
char    canonb[CANBSIZ];        /* buffer for erase and kill (#@) */

#include "sys/map.h"

struct map coremap[CMAPSIZ];    /* space for core allocation */
struct map swapmap[SMAPSIZ];    /* space for swap allocation */
struct map ubmap[13];           /* Unibus Map allocation map */
int     *rootdir;               /* pointer to inode of root directory */
int     *runq;                  /* head of linked list of running processes */
int     cputype;                /* type of cpu =40, 45, or 70 */
int     lbolt;                  /* time of day in 60th not in time */
long    time;                   /* time in sec from 1970 */
long    tout;                   /* time of day of next sleep */
int     *acctp;                 /* inode of accounting file */
struct {
        char    ac_comm[DIRSIZ];        /* Accounting command name */
        char    ac_flag;                /* Accounting flag (unused) */
        char    ac_uid;                 /* Accounting user ID */
        long    ac_date;                /* Accounting start time of command*/
        long    ac_etime;               /* Accounting elapsed time */
        long    ac_utime;               /* Accounting user time */
        long    ac_stime;               /* Accounting system time.*/
        long    ac_dread;               /* Accounting disk reads */
        long    ac_dwrit;               /* Accounting disk writes */
} acctbuf;

/*
 * The callout structure is for
 * a routine arranging
 * to be called by the clock interrupt
 * (clock.c) with a specified argument,
 * in a specified amount of time.
 * Used, for example, to time tab
 * delays on typewriters.
 */
struct  callo
{
        int     c_time;         /* incremental time */
        int     c_arg;          /* argument to routine */
        int     (*c_func)();    /* routine */
} callout[NCALL];
/*
 * Mount structure.
 * One allocated on every mount.
 * Used to find the super block.
 */
struct  mount
{
```

```
        int     m_dev;          /* device mounted */
        int     *m_bufp;        /* pointer to superblock or swap block */
        int     *m_inodp;       /* pointer to mounted on inode */
        int     m_flags;        /* mount flags */
        int     m_refc;         /* Incore reference count */
} mount[NMOUNT];
#define M_RONLY 01
#define M_INCOR 02
#define M_GET   04
#define M_WANT  010

int     mpid;                   /* generic for unique process id's */
char    runin;                  /* scheduling flag */
char    runout;                 /* scheduling flag */
char    runrun;                 /* scheduling flag */
char    curpri;                 /* more scheduling */
char    nxtpri;                 /* more scheduling */
char    runlock;                /* sched flag used in text&data locking */
int     maxmem;                 /* actual max memory per process */
int     memsiz;                 /* max core allowed to be used */
int     *lks;                   /* pointer to clock device */
int     rootdev;                /* dev of root see conf.c */
int     swapdev;                /* dev of swap see conf.c */
int     swplo;                  /* block number of swap space */
int     nswap;                  /* size of swap space */
char    updlock;                /* lock for sync */
char    suincnt;                /* number of superblocks currently in core */
int     rablock;                /* block to be read ahead */
char    regloc[];               /* locs. of saved user registers (trap.c) */
char    msgbuf[MSGBUFS];        /* saved "printf" characters */
char    *ooswap;                /* out of swap space printf message */

/*
 * routine definitions
 */
struct buf *aagetblk();
struct buf *aabread();
struct buf *getablk();

int     dqlen;                  /* number of requests on disk queues */
int     blkacty;                /* active block devices for error log */
int     chracty;                /* active character devices */

/*
 * Instrumentation
 *      NIOSTAT is the number of devices to gather statistics on
 *      and it is defined in param.h
 */
int     dk_busy;
long    dk_time[(4<<NIOSTAT)];
long    dk_numb[NIOSTAT];
long    dk_wds[NIOSTAT];

/* disk seek profiling */
#ifdef CYLHIST
int     dk_unit;
```

```
long    dk_acyl[103];
long    dk_scyl[103];
#endif
```

```
/*      @(#)text.h      2.3     */

/*
 * Text structure.
 * One allocated per pure
 * procedure on swap device.
 * Manipulated by text.c
 */
struct text
{
        int     x_daddr;        /* disk address of segment */
        int     x_caddr;        /* core address, if loaded */
        int     x_size;         /* size (*64) */
        int     *x_iptr;        /* inode of prototype */
        char    x_count;        /* reference count */
        char    x_ccount;       /* number of loaded references */
        char    x_flag;         /* traced, written flags */
        char    x_lcount;       /* number of locked references */
};

#define XTRC    01              /* Text may be written, exclusive use */
#define XWRIT   02              /* Text written into, must swap out */
#define XLOAD   04              /* Currently being read from file */
#define XLOCK   010             /* Being swapped in or out */
#define XWANT   020             /* Wanted for swapping */
```

```
/*      @(#)textx.h     2.3     */

/*
 * Allocation of the text table
 */

struct  text text[NTEXT];
```

```
/*      @(#)timeb.h     2.1     */

/*
 * Structure returned by ftime system call
 */
struct timeb {
        time_t  time;
        unsigned short millitm;
        short   timezone;
        short   dstflag;
};
```

```
/*      @(#)trans.h     2.4.1.1 */

 /* don't alter layout without consulting tty.h */
#ifdef KERNEL
struct  tstp {
        int     ts_flags;
        int     ts_state;
        int     ts_dev;
        char    ts_ltype;
        char    ts_delct;
        char    ts_tout;
        char    ts_dstat;
        int     ts_addr;
        int     ts_speeds;
        struct  clist ts_rawq;
        struct  clist ts_canq;
        struct  clist ts_outq;
        char    ts_erase;
        char    ts_kill;
        char    ts_quanta;
        char    ts_dcnt;
        int     ts_count;
        char    ts_brk0;
        char    ts_brk1;
        int     ts_pgrp;
        struct chan *ts_chan;
        caddr_t ts_linep;
};
#endif

/*
 * ioctl arg structure
 */
struct transcb {
        char    trs_quanta;
        char    trs_fill;
        char    trs_bk0;
        char    trs_bk1;
};

#define TRANS_LTYPE     (short)3
```

```
/*      @(#)tty.h       2.9     */

/*
 * A clist structure is the head
 * of a linked list queue of characters.
 * The characters are stored in 4-word
 * blocks containing a link and 6 characters.
 * The routines getc and putc (m45.s or m40.s)
 * manipulate these structures.
 */
struct clist
{
        int     c_cc;           /* character count */
        int     c_cf;           /* pointer to first char (block) */
        int     c_cl;           /* pointer to last  char (block) */
};

/*
 *      PACKETSIZ defines the number of data bytes in a clist block.
 *              This define must be (2^n)-1. Any change to this define
 *              must be accompanied by a corresponding change to
 *              CBITS in mch.s
 */
#define PACKETSIZ       30

/*
 * The actual structure of a clist block manipulated by
 * getc and putc (mch.s)
 */

struct cblock {
        struct  cblock *c_next;         /* pointer to next block in list */
        char    info[PACKETSIZ];        /* data bytes */
};

/*
 * A tty structure is needed for
 * each UNIX character device that
 * is used for normal terminal IO.
 * The routines in tty.c handle the
 * common code associated with
 * these structures.
 * The definition and device dependent
 * code is in each driver. (kl.c dc.c dh.c)
 * The following define is to keep new mpx code happy.
 */

#define t_line  t_ltype

struct tty
{
        short   t_flags;        /* mode, settable by stty call */
        short   t_state;        /* current state of device */
        short   t_dev;          /* major/minor device no.s of line */
        char    t_ltype;        /* line discipline type number */
```

```
        char    t_delct;        /* number of delimiters in raw q */
        char    t_term;         /* terminal type number */
        char    t_dstat;        /* line state (used by line discipline) */
        caddr_t t_addr;         /* device address (register or startup fcn) */
        short   t_speeds;       /* output+input line speed */
        struct  clist t_rawq;   /* raw input chars directly from device */
        struct  clist t_canq;   /* processed input chars after canon fcn */
        struct  clist t_outq;   /* output character list to device */
        char    t_erase;        /* character delete */
        char    t_kill;         /* line delete */
        char    t_col;          /* current column (0-79) */
        char    t_row;          /* current row */
        char    t_vrow;         /* first variable row */
        char    t_hqcnt;        /* no. of high queue packets in t_outq */
        char    t_lrow;         /* last physical row on the CRT */
        char    t_tmflgs;       /* terminal flags */
        short   t_pgrp;         /* process group name */
        struct chan *t_chan;    /* destination channel */
        caddr_t t_linep;        /* aux line discipline pointer */
};

/*
 * tty DMA output control structure -- used by DH11
 */

struct tty_dma {
        char    *dma_blk;       /* Clist packet pointer */
        int     dma_char;       /* Used for break and timeout */
        int     dma_xmem;       /* Used for extended memory addr */
};

/*
 * Character device priorities
 */

#define TTIPRI  10
#define TTOPRI  20

/*
 * Default special characters
 */

#define CERASE  '#'             /* single character erase */
#define CKILL   '@'             /* erase line */
#define CEOT    004
#define SCERASE '_'             /* single character erase under STDTTY */
#define SCKILL  '$'             /* erase line under STDTTY */
#define XOFF    023             /* suspend dma output */
#define XON     021             /* restart output */
#define XMTESC  033             /* stop/start output */

/*
 * Character error flags for silo devices
 */

#define PERROR  010000          /* parity error */
```

```
#define FRERROR 020000          /* framing error */

/*
 * Hardware dependent defines.
 * Hopefully these are universal.
 */

#define CARRIER 0100
#define RING    0200
#define DONE    0200
#define IENABLE 0100

/*
 * TTY character packet limits
 */

#define TTHIWAT 240
#define TTLOWAT 30
#define TTYHOG  256

/*
 * Device type modes (set in t_flags)
 */

#define MHNLDELAY       01400
#undef  NLDELAY

#define NCDELAY 01              /* no carriage return delay */
#define XTABS   02              /* map tabs to spaces on output */
#define LCASE   04              /* upper case only terminal */
#define ECHO    010             /* echo all received characters */
#define CRMOD   020             /* map CR->LF;echo CR or LF as CR-LF */
#define RAW     040             /* raw character input */
#define ODDP    0100            /* odd parity rcvd/xmtd */
#define EVENP   0200            /* even parity rcvd/xmtd */
#define ANYP    0300            /* any parity mask */
#define HDPLX   0400            /* Half duplex line */
#define NOHUP   01000           /* not dial device flag */
#define XCLUDE  02000           /* disallow future opens */
#define NOSLEEP 04000           /* dont sleep if nothing is ready */
#define NTDELAY 010000          /* no tab delay flag */
#define NLDELAY 020000          /* no newline delay flag */
#define TANDEMI 040000          /* enable response to xon/xoff */
#define STDTTY  0100000         /* non-standard tty escapes and kills */

/*
 * Line/Device state bits (set in t_state)
 */

#define TIMEOUT 01              /* Delay timeout in progress */
#define WOPEN   02              /* Waiting for open to complete */
#define ISOPEN  04              /* Device is open */
#define SSTART  010             /* Has special start routine at addr */
#define CARR_ON 020             /* Software copy of carrier-present */
#define BUSY    040             /* Output in progress */
#define XMTSTOP 0100            /* transmitting suspended */
```

```
#define TBLOCK  0200            /* tandem stop bit */
#define STANDEMO 0400           /* enable transmission of xon/xoff */
#define EVEROPEN 01000          /* line has been opened before */
#define QLOCKB  02000           /* t_outq locked for base level */
#define QLOCKI  04000           /* t_outq locked for interrupts */
#define QWANT   010000          /* base level wants t_outq */
#define INESC   020000          /* Input char '\\' flag */
#define XMTXOFF 040000          /* xmit suspended by XOFF */
#define ASLEEP  0100000         /* is sleeping */

/*
 * Line states as used by hf.c (set in t_dstat).
 */

#define RCV_WAIT        01      /* Receiving temporarily turned off */
#define LNTRNON         02      /* Line being turned on */
#define RQS_ON          04      /* Request to send status */
#define XMT_ON          010     /* Clear to send on */
#define SEC_ON          020     /* Secondary carrier being received */
#define NCHOUT          040     /* No character xmtd since turn around began */
#define REC_ON          0100    /* Receiving carrier from terminal */
#define LNTRNRD         0200    /* Line being turned around */

/*
 * Terminal flags (set in t_tmflgs).
 */

#define SNL     1               /* non-standard new-line needed */
#define ANL     2               /* automatic new-line */
#define LCF     4               /* Special treatment of last col, row */
#define TERM_CTLECHO    010     /* Echo terminal control characters */
#define TERM_INVIS      020     /* do not send escape sequences to user */
#define TM_SET          0200            /* must be on to set/reset flags */
#define TERM_BIT TM_SET         /* Bit reserved for terminal drivers.  */
                                /* Usually used to indicate that an esc*/
                                /* character has arrived and that the  */
                                /* next character is special.          */
                                /* This bit is the same as the TM_SET  */
                                /* bit which may never be set by a user*/

/*
 * Character flags
 */

#define CPRES   0100000         /* valid character present flag */
#define CTOUT   040000          /* timeout flag */
#define CBREAK  020000

/*
 * t_outq special characters;
 * all negative chars following QESC
 * are reserved for timing.
 */

#define QESC    0177
#define QBRK    0
```

```
#define HOEND   1

/*
 * Third argument to line disc sgtty routines indicating if
 * the disc is being turned on (LSET), turned off (LUNSET),
 * or simply having some of its parameters changed (LRESET).
 */

#define LRESET  0
#define LSET    1
#define LUNSET  2

/*
 * following line is a kludge for CB-UNIX issue 2
 */
```

```
/*      @(#)ttyx.h      2.3     */

/*
 * The character lists -- space is allocated in tty.c
 */

struct cblock cfree[];

/*
 * List head for unused character blocks.
 */

struct cblock *cfreelist;

/*
 * ASCII table: parity, character class (defined in partab.c)
 */

char partab[];

/*
 * Character delay table -- number of clock ticks required for a character
 * time at a given speed.  Indexed by tp->t_speed&017.
 */

char chrdelay[];
```

```
#       @(#)tu.mk       2.1


INS = cpmv
FRC =

COMPOOL = /usr/include/util/sys

HEADERS = \
        $(COMPOOL)/param.h

all:    $(FRC) $(HEADERS)
        @echo Headers are now up to date.

$(COMPOOL)/param.h:     param.util.h
        cpmv param.util.h       --664 src sys $@

FRC:
        rm -f $(HEADERS)

.PRECIOUS:      $(HEADERS)


.h~.h:
        get -s $<
```

```
/*      @(#)types.h     2.2     */
/*
 * Typedefs
 */
typedef struct { int r[1]; } *  physadr;
typedef         unsigned        daddr_t;
typedef         char *          caddr_t;
typedef         unsigned int    ino_t;
typedef         long            time_t;
typedef         int             label_t[6];
typedef         int             dev_t;
typedef         long            off_t;
typedef         long            paddr_t;
typedef         unsigned int    spcnt_t;
```

```
/*      @(#)user.h      2.8     */

/*
 * The user structure.
 * One allocated per process.
 * Contains all per process data
 * that doesn't need to be referenced
 * while the process is swapped.
 * The user block is USIZE*64 bytes
 * long; resides at virtual kernel
 * loc 140000; contains the system
 * stack per user; is cross referenced
 * with the proc structure for the
 * same process.
 */

#define EXCLOSE 01

struct user
{
        int     u_rsav[2];              /* save r5,r6 when exchanging stacks */
        int     u_fsav[25];             /* save fp registers */
                                        /* rsav and fsav must be first in
                                           user.h structure */
        char    u_segflg;               /* IO flg: 0:user D; 1:sys; 2:user I */
        char    u_error;                /* return error code */
        char    u_uid;                  /* effective user id */
        char    u_gid;                  /* effective group id */
        char    u_ruid;                 /* real user id */
        char    u_rgid;                 /* real group id */
        int     u_procp;                /* pointer to proc structure */
        char    *u_base;                /* base address for IO */
        unsigned        u_count;                /* bytes remaining for IO */
        off_t   u_offset;               /* offset in file for IO */
        int     *u_cdir;                /* ptr to inode of current directory */
        char    u_dbuf[DIRSIZ];         /* current pathname component */
        char    *u_dirp;                /* current pointer to inode */
        struct  {                       /* current directory entry */
                int     u_ino;
                char    u_name[DIRSIZ];
        } u_dent;
        int     *u_pdir;                /* inode of parent directory of dirp */
        int     u_uisa[16];             /* proto of segmentation addresses */
        int     u_uisd[16];             /* proto of segmentation descriptors */
        int     u_ofile[NOFILE];        /* open file pointers */
        char    u_pofile[NOFILE];       /* per-process flags of open files */
        int     u_arg[5];               /* arguments to current system call */
        int     u_tsize;                /* text size (*64) */
        int     u_dsize;                /* data size (*64) */
        int     u_ssize;                /* stack size (*64) */
        int     u_sep;                  /* flag for I and D separation */
        int     u_qsav[2];              /* label variable for quit and intrp */
        int     u_ssav[2];              /* label variable for swapping */
        int     u_signal[NSIG];         /* disposition of signals */
        time_t  u_utime;                /* this process user time */
```

```
        time_t  u_stime;                /* this process system time */
        time_t  u_cutime;               /* sum of childs' utimes */
        time_t  u_cstime;               /* sum of childs' stimes */
        int     *u_ar0;                 /* address of users saved R0 */
        int     u_prof[4];              /* profile arguments */
        char    u_intflg;               /* interrupted sys call flag */
        char    u_dsleep;               /* scheduling flag */
        int     u_ttyp;                 /* controlling tty pointer */
        int     u_ttyd;                 /* controlling tty dev */
        struct {                        /* header of executable file */
                int     ux_mag;         /* magic number */
                unsigned ux_tsize;      /* text size */
                unsigned ux_dsize;      /* data size */
                unsigned ux_bsize;      /* bss size */
                unsigned ux_ssize;      /* symbol table size */
                unsigned ux_entloc;     /* entry location */
                unsigned ux_unused;     /* relocation base */
                char    ux_relflg;      /* relocation flag */
                char    ux_uver;        /* UNIX version */
        } u_exdata;
        int     *u_rdir;                /* root for this process */
        int     u_gbcnt;                /* number of calls to getblock */
        long    u_dread;                /* number of disk reads */
        long    u_dwrit;                /* number of disk writes */
        long    u_cgbcnt;               /* sum of childs' gbcnts */
        long    u_cdread;               /* sum of childs' dreads */
        long    u_cdwrit;               /* sum of childs' dwrits */
        int  u_narge;                   /* used only during exec */
        struct  {                       /* MAUS mem mgt save area */
                int     ms_uisa;
                int     ms_uisd;
        } u_msav[NMSAV];
        char    u_mbitm;                /* MAUS bitmap--MM Reg utilization */
        char    u_indflg;               /* set if indir sys call */
        int     *u_msgqhdr;             /* ptr to message q header */
        char    u_comm[DIRSIZ];         /* last increment in prog name */
        char    u_acflag;               /* process accounting flag */
        char    u_semflg;               /* semaphore flag byte */
        long    u_start;
        int     u_call;
        int     u_narg;
        int     u_nargc;
        int     u_loc;
        int     u_dev;
        int     u_inode;
        int     u_semav;
        int     u_mask;
        int     u_lock;                 /* text & data lock flag word */
        int     u_stack[1];

                                        /* kernel stack per user
                                         * extends from u + USIZE*64
                                         * backward not to reach here
                                         */
};

/* u_error codes */
```

```
#define EFAULT  106
#include "sys/syserr.h"

/* ioflag values: Read/Write, User/Kernel, Ins/Data */
#define U_WUD   0
#define U_RUD   1
#define U_WKD   2
#define U_RKD   3
#define U_WUI   4
#define U_RUI   5
#define U_WKI   6
#define U_RKI   7
```

```
/*      @(#)userx.h     2.3     */

/*
 * Allocation of the user structure.
 */

struct user u;
```

```
#       @(#)util.mk     2.1

INS = cpmv
FRC =

COMPOOL = /usr/include/util/sys

HEADERS = \
        $(COMPOOL)/param.h

all:    $(FRC) $(HEADERS)
        @echo Headers are now up to date.

$(COMPOOL)/param.h:     param.util.h
        cpmv param.util.h       --664 src sys $@

FRC:
        rm -f $(HEADERS)

.PRECIOUS:      $(HEADERS)

.h~.h:
        get -s $<
```

```
/*      @(#)utsname.h   2.1     */
struct utsname {
        char    sysname[9];
        char    nodename[9];
        char    release[9];
        char    version[9];
};
extern struct utsname utsname;
```

```
/*      @(#)version.h   2.2     */
/*
 * Operating System Version Definitions
 *
 *  These values are stored in u.u_exdata.ux_uver for a.out files.
 *   and indicate which operating system environment the file expects.
 */

#define UV_DEF  0       /* default -- version 6 file system*/
#define UV_CBR2 12      /* CB UNIX Release 2 -- version 6 file system*/
#define UV_CBR3 13

#define VERSION(a) (u.u_exdata.ux_uver == a)

#ifndef KERNEL
/* Stucture accessed by cc to assign stamp versions */

struct unx_ver {
        char cc_ver;
        char un_ver;
} unx_ver[] = {
        'c',  UV_CBR3,          /* cc -- default */
        'o',  UV_CBR2,          /* occ */
};
#endif
```

```
/*      @(#)votrax.h    2.1     */

/*
 * ioctl arg structure
 */
struct vscb {
        char    vot_tab;
        char    vot_flsh;
};

#define VOT_LTYPE       (short)5
```

```
/*      @(#)vt11.h      2.6     */

#
/*
 * VT11 Set Graphic Mode
 */

#define VTMODBTS        0174000         /* mode bit mask */
#define VTCHAR          0100000         /* char mode */
#define VTSHORTV        0104000         /* short vector mode */
#define VTLONGV         0110000         /* long vector mode */
#define VTPOINT         0114000         /* point mode */
#define VTGRAPHX        0120000         /* graphplot x mode */
#define VTGRAPHY        0124000         /* graphplot y mode */
#define VTRELATV        0130000         /* relative point mode */

#define VTINT0  02000                   /* intensity 0 (dimmest) */
#define VTINT1  02200
#define VTINT2  02400
#define VTINT3  02600
#define VTINT4  03000
#define VTINT5  03200
#define VTINT6  03400
#define VTINT7  03600

#define VTLINE0 4                       /* solid line */
#define VTLINE1 5                       /* long  dash */
#define VTLINE2 6                       /* short dash */
#define VTLINE3 7                       /* dot   dash */

#define VTBLNKON        030             /* blink on  */
#define VTBLNKOF        020             /* blink off */

/*
 * Load Status Register A
 */

#define VTSTATSA        0170000         /* load status register a */
#define VTDSTOP         0173400         /* display stop + interrupt */
#define VTSINON         01400           /* stop interrupt on */
#define VTSINOF         01000           /* stop interrupt of */
#define VTLPLITE        0200            /* intensify lp hit */
#define VTLPDARK        0300            /* dont intensify lp hit */
#define VTITAL0         040             /* italics off */
#define VTITAL1         060             /* italics on  */
#define VTSYNC          04              /* halt and resume in sync */

/*
 * Load Status Register B
 */

#define VTSTATSB        0174000         /* load status register b */
#define VTINCR          0100            /* graphplot increment */

/*
```

```
 * Miscellaneous Instructions
 */

#define VTDJMP  0160000                 /* display jump instruction */
#define VTDNOP  0164000                 /* no op instruction */

/*
 * Long Vector and Point Mode Defines
 */

#define VTINTX  040000                  /* intensify vector or point */
#define VTMAXX  01777                   /* maximum x (i.e. mask) */
#define VTMAXY  01777                   /* maximum y (i.e. mask) */
#define VTMINUS 020000                  /* negative x/y component */

/*
 * Short Vector and Relative Point Defines
 */

#define VTMAXSX 017600          /* mask for x */
#define VTMAXSY 077             /* mask for y */
#define VTMISVX 020000          /* - for x in short vectors */
#define VTMISVY 0100            /* - for y in short vectors */

/*
 * VT11 Definitions
 */

#define VTADDR  0172000                 /* Device Address */

#define VTSPRI  10                      /* priority of stop sleeps */
#define VTBPRI  -30                     /* priority of waiting for block */

#define VTNFRM  10                      /* Number of VT11 frames */
#define VTBUFL  256                     /* Length of VT11 frame piece */
#define VTMXBFS 6                       /* Max. # of sys bufs */

/* v_flags */
#define VTGO    1                       /* VT11 running */
#define VTSTOP  2                       /* Please stop VT11 */
#define VTBFREE 4                       /* Frame waiting to be freed */
#define VTUSER  010                     /* User has VT11 opened */
#define VTSYS   020                     /* System has VT11 opened */

/*
 * vtregs - VT11 hardware register layout.
 */

struct vtregs {
        int vtdpc;                      /* display processor address */
        int vtdsr;                      /* display status register */
        int vtxsr;                      /* x / graphplot register */
        int vtysr;                      /* y / character register */
};

/*
```

```
 * vtfrpt - Special structure for passing frame insertion data
 *          to vtwrite from system routines.
 */

struct vtfrpt {
        int vtfrmn;                     /* frame # to be inserted */
        int vtfrad;                     /* address of frame */
        int vtfrcn;                     /* # of bytes in frame */
};

/*
 * vtframp - Used to point to the actual frame from the VT11 master
 *           loop structure.
 */

struct vtframp {
        int vtjump;                     /* filled with VTDJMP's */
        int vtfradr;                    /* address of frame contents */
};

/*
 * vtcycl - VT11 master loop structure.
 */

struct vtcycl {
        int vtstop;                     /* point at which DPU will stop */
        int vtsync;                     /* sync instruction */
        struct vtframp vtjyfr;          /* special joystick frame */
        struct vtframp vtlpfr;          /* special lp frame */
        struct vtframp vtfrm[VTNFRM];   /* regular VT11 frames */
        struct vtframp vtloop;          /* loop back to stop */
};

/*
 * vtframe - Frame header.
 *           NOTE: Do not change without consulting buf.h for the layout
 *                 of buf headers.
 */

#define vtframe buf
#define v_flags b_active                /* devtab association */
#define v_frmpc av_back                 /* Next frame piece */
#define v_addr b_paddr                  /* Frame piece storage pointer */

/*
 * vtfrhd - VT11 frame head place keepers.  Holds pointer to
 *          the system buffer heading a VT11 frame.
 */

struct vtfrhd {
       struct vtframe *vfd_frm;
};
```

```
/*      @(#)vtmn.h      2.4     */

#define PR_CLEAR        0               /* clear code for proclin */
#define PR_BLINK        1               /* blink proc name */
#define PR_STATE        2               /* proc state */
#define PR_FLAG         3               /* proc flag */
#define PR_WCHAN        4               /* proc wchan (sleep) */
#define PR_SIG          5               /* proc caught signal entry */
#define PR_PRI          6               /* proc priority */
#define PR_PTIM         7               /* proc time on current medium */
#define PR_CTIM         8               /* proc time in current state */
#define PR_CLOCK        9               /* proc controlling tty entry */
#define PR_GROUP        10              /* process group */
#define PR_PID          11              /* proc id entry */
#define PR_PPD          12              /* proc parent id entry */
#define PR_NAME         13              /* process name */
#define PR_SWH          14              /* proc going to switch */
#define PR_WKP          15              /* proc waking up */
#define PR_NEW          16              /* new process */
#define PR_PSIG         17              /* signal caught */
#define PR_SWP          18              /* swap call */
#define PR_BNKOF        19              /* no one active */
#define PR_SIZE         20              /* proc size */

#define HD_CLEAR        0               /* clear header line */
#define HD_TYPE         1               /* CPU owner entry */
#define HD_TIME         2               /* current time */
#define HD_IDLE         3               /* idle header */
#define HD_SWPON        4               /* swap in progress */
#define HD_SWPOF        5               /* swap done */
#define HD_BUFON        6               /* buffer in use */
#define HD_BUFOF        7               /* buffer freed */
#define HD_SYSA         8               /* system active */

#ifdef VTMON
#define VTTINC()        vttinc()
#define VTPROCENT(one,two)      vtprocent(one,two)
#define VTNEWPROC(one,two,three,four)   vtprocent(one,two,three,four)
#define VTOPEN()                        vtopen()
#define VTMISCENT(one,two,three)        vtmiscent(one,two,three)
#endif

#ifndef VTMON
#define VTTINC()
#define VTNEWPROC(one,two,three,four)
#define VTPROCENT(one,two)
#define VTOPEN()
#define VTMISCENT(one,two,three)        three;
#endif
```

```
/*      @(#)acct.c      2.5     */

#include        "sys/param.h"
#include        "sys/systm.h"
#include        "sys/user.h"
#include        "sys/userx.h"
#include        "sys/inode.h"

/*
 * Perform process accounting functions.
 */

#ifdef SYSACCT
sysacct()
{
        extern uchar();
        register struct inode *ip;
        register fmt;

        if (suser()) {
                if (u.u_arg[0]==0) {
                        if (acctp) {
                                plock(acctp);
                                iput(acctp);
                                acctp = NULL;
                        }
                        return;
                }
                if (acctp) {
                        u.u_error = EBUSY;
                        return;
                }
                u.u_dirp = u.u_arg[0];
                if ((ip = namei(&uchar, 0))==NULL)
                        return;
                fmt = ip->i_mode & IFMT;
                if((fmt != IFREG) && (fmt != IFLRG)) {
                        u.u_error = EACCES;
                        iput(ip);
                        return;
                }
                acctp = ip;
                prele(ip);
        }
}

/*
 * On exit, write a record on the accounting file.
 */
acct()
{
        register struct inode *ip;
        register i;
        off_t siz;
```