subject: UNIX Operating System - Change to Header Files

date: June 7, 1974

from: T. M. Raleigh

In order to provide an aid to the UNIX Users' Community in reading system code, comments have been added to the system header files. A listing of the files is attached and feedback is requested as to the form and substance of the comments.

It is intended that these commented files will be distributed as an update to the current header files in the near future.

T. M. Raleigh

MH-TMR-dh

Copy to
UNIX Distribution List

TAGUE B 1
MH 2C-126

```
struct buf {
        int     b_flags;                /*see below* ← where below!
        struct  buf *b_forw;            /*doubly linked list of buffers
                                         *for device - block type */
        struct  buf *b_back;            /* " */
        struct  buf *av_forw;           /*doubly linked list of buffers
                                         *eligible for re-allocation */
        struct  buf *av_back;           /* " */
        int     b_dev;                  /*device type to be used in transfer*/
        int     b_wcount;               /* -1 times no. of words to be transfered*/
        char    *b_addr;                /*core address of the buffer, see also B_XMEM*/
        char    *b_blkno[2];            /*block position to be used in transfer*/
        char    b_error;                /*transfer failure reason - currently unused*/
        char    *b_resid;               /*residual word count*/
} buf[NBUF];                            /*table of buffer headers*/

/*
 * forw and back are shared with "buf" struct.
 */
struct devtab {                         /*buffer info per device*/
        char    d_active;               /*set if I/O in progress*/
        char    d_errcnt;               /*count of errors on device*/
        struct  buf *b_forw;            /*head of doubly linked chain of
                                         *idle buffers for this device */
        struct  buf *b_back;            /* " */
        struct  buf *d_actf;            /*head of doubly linked chain of active
                                         *buffers for this device */
        struct  buf *d_actl;            /* " */
};

struct  buf bfreelist;                  /*root of list of reallocatable buffers*/

/* flags */
#define B_WRITE 0                       /*ls bit is zero for write transfer*/
#define B_READ  01                      /*set if read transfer*/
#define B_DONE  02                      /*set when transfer complete*/
#define B_ERROR 04                      /*set if transfer failed for reason b_error*/
#define B_BUSY  010                     /*set if buffer is not eligible
                                         *for re-allocation */
#define B_XMEM  060                     /*mask for high order bits of
                                         *physical core address (b_addr) */
#define B_WANTED 0100                   /*set if someone is sleeping
                                         *for the block header */
#define B_RELOC 0200                    /*unused*/
#define B_ASYNC 0400                    /*set if buffer to be released
                                         *when transfer is complete */
#define B_DELWRI 01000                  /*set if must be written away
                                         *before being re-allocated */
```

```
struct
{
        char    d_minor;        /* minor device code typically logical device
                                 * attached to a controller */
        char    d_major;        /* major device code - displacement into
                                 * conf.c table, one per controller */
};

struct  bdevsw
{
        int     (*d_open)();    /*routine to call on sysopen*/
        int     (*d_close)();   /*routine to call on sysclose*/
        int     (*d_strategy)(); /*routine to call for sysread and syswrite*
        int     *d_tab;         /*pointer to struct devtab for the device*/
} bdevsw[];                     /*one entry for each major device type
                                 * (block devices) */
int     nblkdev;                /*%*/

struct  cdevsw
{
        int     (*d_open)();    /*routine to call on sysopen*/
        int     (*d_close)();   /*routine to call for sysclose*/
        int     (*d_read)();    /*routine to call for sysread*/
        int     (*d_write)();   /*routine to call for syswrite*/
        int     (*d_sgtty)();   /*routine to call for sysgtty and sysstty*/
} cdevsw[];                     /*one entry for each major device type
                                 * (character devices) */
int     nchrdev;                /*%*/
```

```
struct file {
        char    f_flag;         /*indicates how file is open - see below*/
        char    f_count;        /*count of invocations*/
        int     f_inode;        /*pointer to entry in inode table*/
        char    *f_offset[2];   /*current byte position for read or write*/
} file[NFILE];                  /*table of open files*/

/* flags */
#define FREAD   01
#define FWRITE  02
#define FPIPE   04
```

```
struct  filsys                  /*super block = block 1 of volume*/
{
        int     s_isize;        /*number of blocks devoted to i-list
                                 * (list starts in block 2) */
        int     s_fsize;        /*track number of first non-allocatable track*
        int     s_nfree;        /*number of valid pointers in s_free[]*/
        int     s_free[100];    /*unallocated block numbers */
        int     s_ninode;       /*number of free i-numbers listed in s_inode[]
        int     s_inode[100];   /*list of free i-numbers*/
        char    s_flock;        /*set when s_free is locked*/
        char    s_ilock;        /*set when s_inode is locked*/
        char    s_fmod;         /*super block has been altered in core
                                 * but not on disk */
        char    s_only;         /*read only file system */
        int     s_time[2];      /*time of last super block update*/
};
```

```
struct inode {                          /*inode as appears on disk in ilist */
        char    i_flag;                 /*see below*/
        char    i_count;                /*count of invocations*/
        int     i_dev;                  /*device for this inode*/
        int     i_number;               /*i-number for this inode*/
                                        /*the following fields are copied
                                         *from disk copy of inode into core when
                                         *file is accessed */
        int     i_mode;                 /*see below*/
        char    i_nlink;                /*number of links to the file*/
        char    i_uid;                  /*user-ID of owner*/
        char    i_gid;                  /*group-ID associated with this file*/
        char    i_size0;                /*ms byte of 24 bit size*/
        char    *i_size1;               /*ls word of 24 bit size*/
        int     i_addr[8];              /*list of block or track numbers used by file*
                                        /*for a special file i_addr[0] is as follo..
                                        ms byte = major device type
                                        ls byte = minor device within major device typ
} inode[NINODE];                        /*table of active inodes*/

/* flags */
#define ILOCK   01                      /*inode locked*/
#define IUPD    02                      /*inode updated but not written on disk*/
#define IACC    04                      /*inode accessed but date not altered on disk*
                                        /*note - IUPD and IACC are not set every time
                                         * an inode is used! */
#define IMOUNT  010                     /*inode for file on which some other
                                         *file system is mounted */
#define IWANT   020                     /*set if someone is waiting on inode
                                         * for ILOCK to be lifted */
#define ITEXT   040                     /*inode refers to shared text source file*/

/* modes */
#define IALLOC  0100000                 /*inode allocated*/
#define IFMT    060000                  /*mask of type code (zero is normal file)*/
#define IFDIR   040000                  /*directory*/
#define IFCHR   020000                  /*special file - character type*/
#define IFBLK   060000                  /*special file - block (mountable) type*/
#define ILARG   010000                  /*large file - addr[] is list of
                                         *blocks containing ptrs to data */
#define ISUID   04000                   /*set user-ID on execution*/
#define ISGID   02000                   /* set group id on execution */
#define IREAD   0400                    /*readable by owner*/
#define IWRITE  0200                    /*writable by owner*/
#define IEXEC   0100                    /*executable by owner*/
```

```
/*
 * variables
 */

#define NBUF    15              /*max no of 512B system buffers*/
#define NINODE  100             /*max no of incore inodes */
#define NFILE   100             /*max no of entries in system File table*/
                                /* note: should be >= NFILE, limits #
                                 * the number of open files */
#define NMOUNT  5               /*max no of mounted file systems*/
#define MAXMEM  (32*32)         /*max user memory in seg units (32*32*32words
#define SSIZE   20              /*initial user stack size in seg units
                                 * (20*32=640words)*/
#define SINCR   20              /*increment for stack growth in seg units*/
#define NOFILE  15              /*max no of file descriptors per process */
#define CANBSIZ 256             /*canonical buf size for each line typed
                                 * for processing # and @ */
#define CMAPSIZ 100             /*max size of core map table*/
#define SMAPSIZ 100             /*max size of swap map table*/
#define NCALL   20              /*max no of time-outs (courtesy calls)*/
#define NPROC   50              /*max no of active processes in system*/
#define NTEXT   20              /*max no of pure procedures active in system*
#define NCLIST  100             /*max no of Clist char tables
                                 * Each table has 1 word link ptr + 6 chars*/


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

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

/*
 * signals
 * dont change
 */

#define NSIG    13              /*number of signal codes*/
#define         SIGHUP  1       /*hangup*/
#define         SIGINT  2       /*interrupt*/
#define         SIGQIT  3       /*quit*/
#define         SIGINS  4       /*illegal instruction*/
#define         SIGTRC  5       /*trace trap*/
#define         SIGIOT  6       /*IOT instruction*/
#define         SIGEMT  7       /*EMT instruction*/
#define         SIGFPT  8       /*floating point exception*/
#define         SIGKIL  9       /*kill*/
#define         SIGBUS  10      /*bus error*/
```

```
#define         SIGSEG  11      /*segmentation violation*/
#define         SIGSYS  12      /*bad argument to sys call*/

/*
 * fundamental constants
 * cannot be changed
 */

#define USIZE   16              /*size of the U. area per process*/
#define NULL    0               /*%*/
#define NODEV   (-1)            /*%*/
#define ROOTINO 1               /*root inode for all filsys is 1*/
#define DIRSIZ  14              /*1 word for inum 14 chars of name*/
```

```
struct proc {
        char    p_stat;         /*see below*/
        char    p_flag;         /*see below*/
        char    p_pri;          /*priority for this process (-ve is high)*/
        char    p_sig;          /*signal code (see param.h)
                                 *sent to this process*/
        char    p_null;         /*unused*/
        char    p_time;         /*seconds of execution since process
                                 *was last swapped in */
        int     p_ttyp;         /*pointer to tty struct for
                                 * originating typewriter */
        int     p_pid;          /*ID of this process*/
        int     p_ppid;         /*ID of parent process*/
        int     p_addr;         /*current position of this process swap area*
                                 /*if swapped in - is core addr (64 byte units
                                 if swapped out addr is block addr on disk*/
        int     p_size;         /*size of process space (64 byte units)*/
        int     p_wchan;        /*reason for this process sleeping*/
        int     *p_textp;       /*pointer to text struct if this process
                                 *uses a shared text segment*/
} proc[NPROC];                  /*one entry in core for each process*/

/* stat codes */
#define SSLEEP  1               /*set if sleeping with low priority*/
#define SWAIT   2               /*set if sleeping with high priority
                                 * (cannot interrupt or kill) */
#define SRUN    3               /*process can run*/
#define SIDL    4               /*unused*/
#define SZOMB   5               /*process has done sysexit*/

/* flag codes */
#define SLOAD   01              /*process swapped in*/
#define SSYS    02              /*set for the scheduling process only*/
#define SLOCK   04              /*set to inhibit consideration by scheduling
                                 * process for swap out */
#define SSWAP   010             /*set if R5,R6 to be restored from u_ssave
                                 * after swap in */
                                /*otherwise restore from u_rsave*/
```

```
char    canonb[CANBSIZ];        /*tty workspace for erase and kill processing
int     coremap[CMAPSIZ];       /*allocation information for core.
                                 * Word pairs as follows:
                                first word is core addr (64 byte units) of ho
                                second word is size of hole (64 byte units)*/
int     swapmap[SWAPSIZ];       /*allocation information for disc swap area.
                                 * Word pairs as follows:
                                first word is disc addr (block) of hole
                                second word is number of blocks in hole*/
int     *rootdir;               /*ptr to inode struct of the root directory*/
int     lbolt;                  /*wait on this for 4-second wakeup*/
                                /*contains counter of 60ths of second*/
int     time[2];                /*current time in seconds*/
int     tout[2];                /*time to wake up those who wait on this loc*/
struct  callo
{
        int     c_time;         /*residual delay (60ths of second)*/
                                /*the delay for the callout[i] is the time
                                 *at which a wakeup should be done
                                 *after callout[i-1] has been awakened*/
        int     c_arg;          /*arg for func to be called when delay ends*/
        int     (*c_func)();    /*function to be called when delay ends*/
} callout[NCALL];               /*one entry for each timed action
                                 * to take place after a given delay */

                                /*mount table one per mounted filsys*/
struct  mount
{
        int     m_dev;          /*dev code  on which file system mounted*/
        int     *m_bufp;        /*pointer to buf struct of the super block*/
        int     *m_inodp;       /*pointer to inode struct with IMOUNT set*/
} mount[NMOUNT];                /*one entry for each mounted file system*/

int     mpid;                   /*next available process number*/
char    runin;                  /*scheduler waits upon these*/
char    runout;                 /* " */
char    runrun;                 /* " */
int     maxmem;                 /*max memory being used by the running system
int     *lks;                   /*device address of the clock
                                 *used by running system */
int     rootdev;                /*dev code for root device*/
int     swapdev;                /*dev code for swap device*/
int     swplo;                  /*lower bound of swap space on swap device*/
int     nswap;                  /*number of blocks in swap space*/
int     updlock;                /*set when syssync being done*/
```

```
/* registers as saved relative to *u_ar0 */
#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)
```

```
/*
 *      Copyright 1973 Bell Telephone Laboratories Inc
 */

/*
 * KT-11 registers
 */

#define KISA    0172340
#define UISD    0177600
#define UISA    0177640
#define R0      02
#define RN      06
#define W0      04
#define ED      010
struct {
        int     r[8];
};
```

```
struct text
{
        int     x_daddr;        /*disc addr on swap device*/
        int     x_caddr;        /*core addr (64 byte units)*/
        int     x_size;         /*size of text segment (64 byte units)*/
        int     *x_iptr;        /*inode for text*/
        char    x_count;        /*number of invokations of the  text*/
        char    x_ccount;       /*no. of swapped in processes that invoked it
} text[NTEXT];                  /*one entry per shared text*/
```

```
#define ENOEXEC 8               /*exec format error*/
#define EBADF   9               /*bad file number*/
#define ECHILD  10              /*no children*/
#define EAGAIN  11              /*no more processes*/
#define ENOMEM  12              /*not enough core*/
#define EACCES  13              /*permission denied*/
#define ENOTBLK 15              /*block device required*/
#define EBUSY   16              /*mount device busy*/
#define EEXIST  17              /*file exists*/
#define EXDEV   18              /*cross-device link*/
#define ENODEV  19              /*no such device*/
#define ENOTDIR 20              /*not a directory*/
#define EISDIR  21              /*is a directory*/
#define EINVAL  22              /*invalid argument*/
#define ENFILE  23              /*file table overflow*/
#define EMFILE  24              /*too many open files*/
#define ENOTTY  25              /*not a typewriter*/
#define ETXTBSY 26              /*text file busy*/
#define EFBIG   27              /*file too large*/
#define ENOSPC  28              /*no space left on device*/
#define ESPIPE  29              /*seek on pipe*/
#define EROFS   30              /*attempted to write on read-only filsys*/
```

```
struct user {
        int     u_rsav[2];      /* U. area */
                                /*R5,R6 save area (switching processes)*/
                                /* must be first */
        int     u_fsav[25];     /* must be second  floating registers */
        char    u_segflg;       /*if set u_base refers to system space,
                                 *otherwise user space*/
        char    u_error;        /*error code - see below*/
        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 struct proc for this process*/
        char    *u_base;        /*addr for current transfer - see u_segflg*/
        char    *u_count;       /*count for current transfer*/
        char    *u_offset[2];   /*offset for current transfer*/
        int     *u_cdir;        /*ptr to inode struct for current directory*/
        char    u_dbuf[DIRSIZ]; /*workspace for namei()*/
        char    *u_dirp;        /* " */
        struct {                /*copy of current directory entry*/
                int     u_ino;  /*i number */
                char    u_name[DIRSIZ]; /*file name in directory */
                }u_dent;
        int     *u_pdir;        /*pointer to inode struct for
                                 *parent directory */
        int     u_uisa[8];      /*segmentation register values
                                 *relative to user base */
        int     u_uisd[8];      /* " */
        int     u_ofile[NOFILE]; /*pointers to file table
                                 *entries for open files */
        int     u_arg[5];       /*system call args*/
        int     u_tsize;        /*text size (64 byte units)*/
        int     u_dsize;        /*data size (64 byte units)*/
        int     u_ssize;        /*stack size (64 byte units)*/
        int     u_qsav[2];      /*R5, R6 save area (after signal)*/
        int     u_ssav[2];      /*R5, R6 save area (while swapping)*/
        int     u_signal[NSIG]; /*on-condition for signals (see param.h)*/
        int     u_utime;        /*execution time for user (60ths of second)*/
        int     u_stime;        /*execution time for system (60ths of second)*
        int     u_cutime[2];    /*cum exec time for user (60ths of second)*/
        int     u_cstime[2];    /*cum exec time for system (60ths of second)*/
        int     *u_ar0;         /*pointer to  register save area (see reg.h)*/
        int     u_prof[4];      /*args for profile*/
        char    u_nice;         /*value to be added to user priority*/
        char    u_dsleep;       /*number of system calls executed
                                 *since last call on sleep() */
                                /*per-process data swapped with the user*/
} u;                            /* u = 140000 */

/* u_error codes                (values >= 100 are fatal) */
#define EFAULT  106             /*%*/
#define EPERM   1               /*not owner and not super-user*/
#define ENOENT  2               /*no such file or directory*/
#define ESRCH   3               /*no such process*/
#define EIO     5               /*I/O error*/
#define ENXIO   6               /*no such device or address*/
#define E2BIG   7               /*arg list too long*/
```