Process Information for UNIX V6

Process Table Entry

This is the process table entry structure. There can be NPROC processes, so the table is of static size. This entry is always in core, even if the process is not running or is swapped out.

```
struct  proc
{
    char    p_stat;     /* process status */
    char    p_flag;     /* process status attributes */
    char    p_pri;      /* priority, negative is high */
    char    p_sig;      /* signal number sent to this process */
    char    p_uid;      /* user id, used to direct tty signals */
    char    p_time;     /* resident time for scheduling */
    char    p_cpu;      /* cpu usage for scheduling */
    char    p_nice;     /* nice for scheduling */
    int     p_ttyp;     /* controlling tty */
    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 */
} proc[NPROC];

/* stat codes */
#define SSLEEP  1       /* sleeping on high priority */
#define SWAIT   2       /* sleeping on low priority */
#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 */
```

Other Part of the Process Information

This is the remainder of the information about the process. It is kept in another area associated with the process. It need not stay in core when the process is swapped out to disk.

```
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 structure */
    char    u_segflg;        /* flag for IO; user or kernel space */
    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 */
    char    *u_count;        /* bytes remaining for IO */
    char    *u_offset[2];    /* offset in file for IO */
    int     *u_cdir;         /* pointer 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;           /* inode number */
        char u_name[DIRSIZ];/* name of directory */
    } u_dent;
    int     *u_pdir;         /* inode of parent directory of dirp */
    int     u_uisa[16];      /* prototype of segmentation addresses */
    int     u_uisd[16];      /* prototype of segmentation descriptors */
    int     u_ofile[NOFILE];/* pointers to file structures 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 quits and interrupts */
    int     u_ssav[2];       /* label variable for swapping */
    int     u_signal[NSIG];  /* disposition of signals */
    int     u_utime;         /* this process user time */
    int     u_stime;         /* this process system time */
    int     u_cutime[2];     /* sum of childs' utimes */
    int     u_cstime[2];     /* sum of childs' stimes */
    int     *u_ar0;          /* address of users saved R0 */
    int     u_prof[4];       /* profile arguments */
    char    u_intflg;        /* catch intr from sys */
                             /* kernel stack per user
                              * extends from u + USIZE*64
                              * backward not to reach here
                              */
} u;
```