OS9-Prozess-Threaddescriptor-Aufbau

```
-------------------------------------------Auszug aus process.h------------------------------------------
....

/* Thread Execution Block */
typedef struct thread {
       u_int32
              t_id;               /* reserved                          */
       process_id
              t_proc;             /* owner process ID (zero if none)                    */
       u_int32
              t_msiz;             /* thread block memory size                                  */
       owner_id
              t_owner;            /* owner's group/user number                                 */
       struct thread
              *t_next,            /* next thread in doubly linked list                         */
              *t_prev,            /* previous thread in doubly linked list              */
              *t_linkn,           /* next list of associated threads (owner link)    */
              *t_linkp;           /* previous list of associated threads (owner link)*/
       u_int32
              t_cycle,            /* wakeup cycle period                                       */
              t_wktime,           /* wakeup time                                               */
              t_flags;            /* thread block flags                                        */
       u_int32
              (*t_function)(); /* function to execute                                         */
       void
              *t_func_pb;         /* parameter block to pass to function                */
} thread, *Thread;


typedef struct prdsc *Pr_desc,pr_desc;

struct prdsc {
       u_int32
              p_sync;            /* process descriptor sync code      */
       process_id
              p_id,              /* process id                 */
              p_pid;             /* parent's process id        */
       owner_id
              p_owner;           /* group/user numbers         */
       u_int32
              p_rsrv1[4];        /* reserved space             */
       Mh_exec
              p_pmodul;          /* primary module pointer     */
       u_int32
              p_pdsize;          /* size of process descriptor */
       u_int16
              p_prior,           /* priority                   */
              p_age;             /* age                        */
       u_int32
              p_sched,           /* process scheduling constant*/
              p_state,           /* process status bit flags   */
              p_queueid,         /* current queue id                  */
              p_preempt,         /* system-state preemption flag, 0 = switchable           */
              p_srstat,          /* process service request capability status       */
              p_srmask;          /* process service request mask      */
       error_code
              p_status;          /* exit (error) status of condemned process               */
       u_int32
              p_timbeg,          /* time when forked in seconds from system ref. date*/
              p_uticks,          /* user state ticks elapsed                         */
              p_sticks;          /* system state tick elapsed                                     */
       process_id
              p_dlockn;          /* process id of next process in deadlock chain           */
       Pr_desc
              p_queuen,          /* general purpose queue next pointer                            */
              p_queuep;          /* general purpose queue previous pointer                 */
       Pr_desc
              p_lockqn,          /* next process in resource lock queue                           */
              p_lockqp;          /* previous process in resource lock queue                */
       Regs
              p_sp;              /* system stack pointer                        */
       u_char
              *p_usp,            /* user stack pointer                          */
              *p_excpsp;         /* system state exception recovery stack                  */
       u_int32
              *p_excppc;         /* system state exception recovery pc                            */
       u_int32
              *p_tvalue,         /* real-time IRQ supporte test value                      */
              *p_taddr;          /* real-time IRQ support test value address               */
       void
              *p_rtistate;   /* pointer to restoration state upon rti trigger    */
       u_char
```

```
        *p_spuimg;            /* pointer to process' SPU image                                  */
    Mod_dir
        p_mdir,               /* process' current module directory                      */
        p_altmdir,            /* process' alternate module directory                            */
        p_smdir;              /* process' current shared module directory       */
    u_int16
        p_sigflg,             /* signal flag                                                    */
        p_siglvl;             /* signal interrupt level                                         */
    u_int32
        (*p_sigvec)(); /* signal intercept vector                                               */
    u_char
        *p_sigdat;            /* signal intercept data address                                  */
    u_int32
        p_sigmask,            /* mask to disable signals 2-31                                   */
        p_sigcnt,             /* number of signals pending                                      */
        p_sigiret,            /* signal intercept recursion counter                             */
        p_siglst,             /* last signal the process received                               */
        p_sigsiz,             /* maximum number of queued signals for process           */
        *p_sigque,            /* pointer to head of signal code queue block             */
        *p_sigqend,           /* pointer to end of signal code queue block      */
        *p_sigin,             /* pointer to next signal "in" slot                               */
        *p_sigout;            /* pointer to next signal "out" slot                      */
    void
        *p_iopd;              /* pointer to associated I/O process descriptor           */
    u_int32
        p_rsrv2[4];           /* reserved space                                         */

#ifndef _RPTHREAD          /* representative process thread ifdef                       */
    u_int32
        p_scall,              /* last user state system call executed                           */
        p_fcalls,             /* number of system calls executed                                */
        p_icalls;             /* number of I/O calls executed                                   */
    u_char
        *p_data;              /* pointer to process' primary data area                  */
    u_int32
        p_datasz;             /* size of primary data area                                      */
    Mem_color
        p_frag[2];            /* process memory fragment list                           */
    u_int32
        *p_miblks,            /* pointer to list of memory image blocks                 */
        *p_memimg[32*2],/* allocated memory block pointers & sizes              */
        *p_pmiblks;           /* allocated & protectected memory pointers & sizes */
    Fpu_desc
        p_fpusave;            /* pointer to FPU save area                               */
    u_int32
        p_chldcnt,            /* count of process' children                                     */
        p_chldrn[(CHILDREN*CHILD_SLOTSZ)+2]; /* child status block              */
    u_int32
        p_evcnt,                          /* number of linked events            */
        p_evtbl[(EVENTS*EV_SLOTS)+2];/* event mapping table (linked events)     */
    Thread
        p_thread[2];                  /* doubly linked system thread queue head */
    u_int32
        *p_except[TRAP_TTL];  /* program error exception vectors                */
    u_char
        *p_exstk[TRAP_TTL];           /* program error exception stack frame ptrs */

#ifndef _MPFPOWERPC
    u_int32
        *p_fpexcpt[FTRAP_TTL];        /* FPU exception vectors                                  */
    u_char
        *p_fpexstk[FTRAP_TTL];        /* FPU exception stack frame pointers             */
#endif /* _MPFPOWERPC */

    Mh_com
        p_sublib[SUBMAX];             /* subroutine module pointers                     */
    u_char
        *p_submem[SUBMAX];            /* subroutine library static memory pointers*/
    Mh_trap
        p_traps[TRAPMAX];             /* user's trap vector table                               */
    u_char
        *p_trpmem[TRAPMAX];           /* user's trap static memory block pointers */
    u_int32
        p_trpsiz[TRAPMAX];            /* trap handler static memory sizes               */
    Pr_desc
        p_dbgpar;                             /* debugging parent process pointer       */
    u_int32
        p_dbgmode,                            /* debug process execution mode
    */
        p_dbgins;                             /* debug execution instruction count      */
    Regs
        p_dbgreg;                             /* debug process register stack frame     */
    Fregs
        p_dbgfreg;                            /* debug process FPU register stack frame */
#if defined(_MPFPOWERPC) || defined(_MPFMIPS) || defined(_MPFARM) || defined(_MPFSPARC)
    u_int32
#else
```

```
        u_int16
#endif /* _MPFPOWERPC */
                p_bpvalue[BRKPTS];            /* breakpoint instruction save area                 */
        void
                *p_dbgrsc;                            /* additional debugging resources (reserved)*/

#endif /* !_RPTHREAD */
        u_int32
                p_rsrv3[7];                           /* reserved space                                   */
        u_char
                p_procstk[STACKSIZE], /* system state stack for process                   */
                p_stackend;
};
/* Process Queue ID codes */
#define Q_NONE        ' '           /* not in any queue                                         */
#define Q_DEAD        '-'     /* no queue: dead process                                 */
#define Q_ACTIVE      'a'           /* active process queue                                     */
#define Q_DEBUG               'd'           /* no queue: inactively debugging                   */
#define Q_EVENT               'e'           /* event queue
        */
#define Q_SLEEP               's'           /* sleep queue
        */
#define Q_WAIT        'w'     /* waiting queue                                                  */
#define Q_CURRNT      '*'     /* no queue: currently running                            */
#define Q_SUSPEND     'z'           /* no queue: state saved and inactive               */
#define Q_SEMA        'p'           /* process is suspended in semaphore queue  */
#define Q_REMOTE      'r'           /* remote processor request wait queue              */
```