```c
#ifndef PROC_H
#define PROC_H

/* Here is the declaration of the process table.  It contains all process
 * data, including registers, flags, scheduling priority, memory map,
 * accounting, message passing (IPC) information, and so on.
 *
 * Many assembly code routines reference fields in it.  The offsets to these
 * fields are defined in the assembler include file sconst.h.  When changing
 * struct proc, be sure to change sconst.h to match.
 */
#include <minix/com.h>
#include "const.h"
#include "priv.h"

struct proc {
  struct stackframe_s p_reg;    /* process' registers saved in stack frame */
  struct segframe p_seg;        /* segment descriptors */
  proc_nr_t p_nr;               /* number of this process (for fast access) */
  struct priv *p_priv;          /* system privileges structure */
  short p_rts_flags;            /* process is runnable only if zero */
  short p_misc_flags;           /* flags that do not suspend the process */

  char p_priority;              /* current scheduling priority */
  char p_max_priority;          /* maximum scheduling priority */
  char p_ticks_left;            /* number of scheduling ticks left */
  char p_quantum_size;          /* quantum size in ticks */

  struct mem_map p_memmap[NR_LOCAL_SEGS];   /* memory map (T, D, S) */

  clock_t p_user_time;          /* user time in ticks */
  clock_t p_sys_time;           /* sys time in ticks */
  clock_t p_recent_time;        /* recent time in ticks */

  struct proc *p_nextready;     /* pointer to next ready process */
  struct proc *p_caller_q;      /* head of list of procs wishing to send */
  struct proc *p_q_link;        /* link to next proc wishing to send */
  message *p_messbuf;           /* pointer to passed message buffer */
  int p_getfrom_e;              /* from whom does process want to receive? */
  int p_sendto_e;               /* to whom does process want to send? */

  sigset_t p_pending;           /* bit map for pending kernel signals */

  char p_name[P_NAME_LEN];      /* name of the process, including \0 */

  endpoint_t p_endpoint;        /* endpoint number, generation-aware */

#if DEBUG_SCHED_CHECK
  int p_ready, p_found;
#endif
};

/* Bits for the runtime flags. A process is runnable iff p_rts_flags == 0. */
#define SLOT_FREE       0x01    /* process slot is free */
#define NO_PRIORITY     0x02    /* process has been stopped */
#define SENDING         0x04    /* process blocked trying to send */
#define RECEIVING       0x08    /* process blocked trying to receive */
#define SIGNALED        0x10    /* set when new kernel signal arrives */
#define SIG_PENDING     0x20    /* unready while signal being processed */
#define P_STOP          0x40    /* set when process is being traced */
#define NO_PRIV         0x80    /* keep forked system process from running */
#define NO_ENDPOINT    0x100    /* process cannot send or receive messages */

/* These runtime flags can be tested and manipulated by these macros. */

```