

```

/* Prototypes for system library functions. */

#ifndef _SYSLIB_H
#define _SYSLIB_H

#ifndef _TYPES_H
#include <sys/types.h>
#endif

#ifndef _IPC_H
#include <minix/ipc.h>
#endif

#ifndef _DEVIO_H
#include <minix/devio.h>
#endif

/* Forward declaration */
struct reg86u;

#define SYSTASK SYSTEM

/*-----*
 * Minix system library. *
 *-----*/
_PROTOTYPE( int _taskcall, (int who, int syscallnr, message *msgptr));

_PROTOTYPE( int sys_abort, (int how, ...));
_PROTOTYPE( int sys_enable_iop, (int proc));
_PROTOTYPE( int sys_exec, (int proc, char *ptr,
                          char *aout, vir_bytes initpc));
_PROTOTYPE( int sys_fork, (int parent, int child, int *));
_PROTOTYPE( int sys_newmap, (int proc, struct mem_map *ptr));
_PROTOTYPE( int sys_exit, (int proc));
_PROTOTYPE( int sys_trace, (int req, int proc, long addr, long *data_p));

_PROTOTYPE( int sys_privctl, (int proc, int req, int i, void *p));
_PROTOTYPE( int sys_nice, (int proc, int priority));

_PROTOTYPE( int sys_int86, (struct reg86u *reg86p));
_PROTOTYPE( int sys_vm_setbuf, (phys_bytes base, phys_bytes size,
                              phys_bytes high));
_PROTOTYPE( int sys_vm_map, (int proc_nr, int do_map,
                              phys_bytes base, phys_bytes size, phys_bytes offset));

/* Shorthands for sys_sdevio() system call. */
#define sys_insb(port, proc_nr, buffer, count) \
    sys_sdevio(DIO_INPUT, port, DIO_BYTE, proc_nr, buffer, count)
#define sys_insw(port, proc_nr, buffer, count) \
    sys_sdevio(DIO_INPUT, port, DIO_WORD, proc_nr, buffer, count)
#define sys_outsb(port, proc_nr, buffer, count) \
    sys_sdevio(DIO_OUTPUT, port, DIO_BYTE, proc_nr, buffer, count)
#define sys_outsw(port, proc_nr, buffer, count) \
    sys_sdevio(DIO_OUTPUT, port, DIO_WORD, proc_nr, buffer, count)
_PROTOTYPE( int sys_sdevio, (int req, long port, int type, int proc_nr,
                             void *buffer, int count));

```

```

/* Clock functionality: get system times or (un)schedule an alarm call.*/
_PROTOTYPE( int sys_times, (int proc_nr, clock_t *ptr));
_PROTOTYPE(int sys_setalarm, (clock_t exp_time, int abs_time));

/* Shorthands for sys_irqctl() system call.*/
#define sys_irqdisable(hook_id) \
    sys_irqctl(IRQ_DISABLE, 0, 0, hook_id)
#define sys_irqenable(hook_id) \
    sys_irqctl(IRQ_ENABLE, 0, 0, hook_id)
#define sys_irqsetpolicy(irq_vec, policy, hook_id) \
    sys_irqctl(IRQ_SETPOLICY, irq_vec, policy, hook_id)
#define sys_irqrmpolicy(irq_vec, hook_id) \
    sys_irqctl(IRQ_RMPOLICY, irq_vec, 0, hook_id)
_PROTOTYPE ( int sys_irqctl, (int request, int irq_vec, int policy,
    int *irq_hook_id) );

/* Shorthands for sys_vircopy() and sys_physcopy() system calls.*/
#define sys_biosin(bios_vir, dst_vir, bytes) \
    sys_vircopy(SELF, BIOS_SEG, bios_vir, SELF, D, dst_vir, bytes)
#define sys_biosout(src_vir, bios_vir, bytes) \
    sys_vircopy(SELF, D, src_vir, SELF, BIOS_SEG, bios_vir, bytes)
#define sys_datacopy(src_proc, src_vir, dst_proc, dst_vir, bytes) \
    sys_vircopy(src_proc, D, src_vir, dst_proc, D, dst_vir, bytes)
#define sys_textcopy(src_proc, src_vir, dst_proc, dst_vir, bytes) \
    sys_vircopy(src_proc, T, src_vir, dst_proc, T, dst_vir, bytes)
#define sys_stackcopy(src_proc, src_vir, dst_proc, dst_vir, bytes) \
    sys_vircopy(src_proc, S, src_vir, dst_proc, S, dst_vir, bytes)
_PROTOTYPE(int sys_vircopy, (int src_proc, int src_seg, vir_bytes src_vir,
    int dst_proc, int dst_seg, vir_bytes dst_vir, phys_bytes bytes));

#define sys_abscopy(src_phys, dst_phys, bytes) \
    sys_physcopy(NONE, PHYS_SEG, src_phys, NONE, PHYS_SEG, dst_phys, bytes)
_PROTOTYPE(int sys_physcopy, (int src_proc, int src_seg, vir_bytes src_vir,
    int dst_proc, int dst_seg, vir_bytes dst_vir, phys_bytes bytes));
_PROTOTYPE(int sys_memset, (unsigned long pattern,
    phys_bytes base, phys_bytes bytes));

/* Vectored virtual / physical copy calls.*/
#if DEAD_CODE /* library part not yet implemented*/
_PROTOTYPE(int sys_virvcopy, (phys_cp_req *vec_ptr,int vec_size,int *nr_ok));
_PROTOTYPE(int sys_physvcopy, (phys_cp_req *vec_ptr,int vec_size,int *nr_ok));
#endif

_PROTOTYPE(int sys_umap, (int proc_nr, int seg, vir_bytes vir_addr,
    vir_bytes bytes, phys_bytes *phys_addr));
_PROTOTYPE(int sys_segctl, (int *index, ul6_t *seg, vir_bytes *off,
    phys_bytes phys, vir_bytes size));

/* Shorthands for sys_getinfo() system call.*/
#define sys_getkmessages(dst) sys_getinfo(GET_KMESSAGES, dst, 0,0,0)
#define sys_getkinfo(dst) sys_getinfo(GET_KINFO, dst, 0,0,0)
#define sys_getloadinfo(dst) sys_getinfo(GET_LOADINFO, dst, 0,0,0)
#define sys_getmachine(dst) sys_getinfo(GET_MACHINE, dst, 0,0,0)
#define sys_getproctab(dst) sys_getinfo(GET_PROCTAB, dst, 0,0,0)
#define sys_getprivtab(dst) sys_getinfo(GET_PRIVTAB, dst, 0,0,0)
#define sys_getproc(dst,nr) sys_getinfo(GET_PROC, dst, 0,0, nr)
#define sys_getrandomness(dst) sys_getinfo(GET_RANDOMNESS, dst, 0,0,0)

```

```

#define sys_getimage(dst) sys_getinfo(GET_IMAGE, dst, 0,0,0)
#define sys_getirqhooks(dst) sys_getinfo(GET_IRQHOOKS, dst, 0,0,0)
#define sys_getirqactids(dst) sys_getinfo(GET_IRQACTIDS, dst, 0,0,0)
#define sys_getmonparams(v,vl) sys_getinfo(GET_MONPARAMS, v,vl, 0,0)
#define sys_getschedinfo(v1,v2) sys_getinfo(GET_SCHEDINFO, v1,0, v2,0)
#define sys_getlocktimings(dst) sys_getinfo(GET_LOCKTIMING, dst, 0,0,0)
#define sys_getbiosbuffer(virp, sizep) sys_getinfo(GET_BIOSBUFFER, virp, \
    sizeof(*virp), sizep, sizeof(*sizep))
/* CODIGO PRACTICA 3 */
#define sys_getmessgsn(dst) sys_getinfo(GET_MESSGSN, dst, 0,0,0)

_PROTOTYPE(int sys_getinfo, (int request, void *val_ptr, int val_len,
    void *val_ptr2, int val_len2) );

/* Signal control. */
_PROTOTYPE(int sys_kill, (int proc, int sig) );
_PROTOTYPE(int sys_sigsend, (int proc_nr, struct sigmsg *sig_ctxt) );
_PROTOTYPE(int sys_sigreturn, (int proc_nr, struct sigmsg *sig_ctxt) );
_PROTOTYPE(int sys_getksig, (int *k_proc_nr, sigset_t *k_sig_map) );
_PROTOTYPE(int sys_endksig, (int proc_nr) );

/* NOTE: two different approaches were used to distinguish the device I/O
 * types 'byte', 'word', 'long': the latter uses #define and results in a
 * smaller implementation, but loses the static type checking.
 */
_PROTOTYPE(int sys_voutb, (pvb_pair_t *pvb_pairs, int nr_ports) );
_PROTOTYPE(int sys_voutw, (pvw_pair_t *pvw_pairs, int nr_ports) );
_PROTOTYPE(int sys_voutl, (pvl_pair_t *pvl_pairs, int nr_ports) );
_PROTOTYPE(int sys_vinb, (pvb_pair_t *pvb_pairs, int nr_ports) );
_PROTOTYPE(int sys_vinw, (pvw_pair_t *pvw_pairs, int nr_ports) );
_PROTOTYPE(int sys_vinl, (pvl_pair_t *pvl_pairs, int nr_ports) );

/* Shorthands for sys_out() system call. */
#define sys_outb(p,v) sys_out((p), (unsigned long) (v), DIO_BYTE)
#define sys_outw(p,v) sys_out((p), (unsigned long) (v), DIO_WORD)
#define sys_outl(p,v) sys_out((p), (unsigned long) (v), DIO_LONG)
_PROTOTYPE(int sys_out, (int port, unsigned long value, int type) );

/* Shorthands for sys_in() system call. */
#define sys_inb(p,v) sys_in((p), (v), DIO_BYTE)
#define sys_inw(p,v) sys_in((p), (v), DIO_WORD)
#define sys_inl(p,v) sys_in((p), (v), DIO_LONG)
_PROTOTYPE(int sys_in, (int port, unsigned long *value, int type) );

/* pci.c */
_PROTOTYPE( void pci_init, (void) );
_PROTOTYPE( void pci_initl, (char *name) );
_PROTOTYPE( int pci_first_dev, (int *devindp, u16_t *vidp, u16_t *didp) );
_PROTOTYPE( int pci_next_dev, (int *devindp, u16_t *vidp, u16_t *didp) );
_PROTOTYPE( int pci_find_dev, (U8_t bus, U8_t dev, U8_t func,
    int *devindp) );
_PROTOTYPE( void pci_reserve, (int devind) );
_PROTOTYPE( void pci_ids, (int devind, u16_t *vidp, u16_t *didp) );
_PROTOTYPE( void pci_rescan_bus, (U8_t busnr) );
_PROTOTYPE( u8_t pci_attr_r8, (int devind, int port) );
_PROTOTYPE( u16_t pci_attr_r16, (int devind, int port) );
_PROTOTYPE( u32_t pci_attr_r32, (int devind, int port) );

```

```
_PROTOTYPE( void pci_attr_w8, (int devind, int port, U8_t value) );  
_PROTOTYPE( void pci_attr_w16, (int devind, int port, U16_t value) );  
_PROTOTYPE( void pci_attr_w32, (int devind, int port, u32_t value) );  
_PROTOTYPE( char *pci_dev_name, (U16_t vid, U16_t did) );  
_PROTOTYPE( char *pci_slot_name, (int devind) );
```

```
#endif /*_SYSLIB_H*/
```