```
1  =============== include/minix/syslib.h ===============
2  /* Prototypes for system library functions. */
3
4  #ifndef _SYSLIB_H
5  #define _SYSLIB_H
6
7  #ifndef _TYPES_H
8  #include <sys/types.h>
9  #endif
10
11 #ifndef _IPC_H
12 #include <minix/ipc.h>
13 #endif
14
15 #ifndef _DEVIO_H
16 #include <minix/devio.h>
17 #endif
18
19 /* Forward declaration */
20 struct reg86u;
21
22 #define SYSTASK SYSTEM
23
24 /*==========================================================================*
25  * Minix system library.                                                    *
26  *==========================================================================*/
27 _PROTOTYPE( int _taskcall, (int who, int syscallnr, message *msgptr));
28
29 _PROTOTYPE( int sys_abort, (int how, ...));
30 _PROTOTYPE( int sys_enable_iop, (int proc));
31 _PROTOTYPE( int sys_exec, (int proc, char *ptr,
32                            char *aout, vir_bytes initpc));
33 _PROTOTYPE( int sys_fork, (int parent, int child));
34 _PROTOTYPE( int sys_newmap, (int proc, struct mem_map *ptr));
35 _PROTOTYPE( int sys_exit, (int proc));
36 _PROTOTYPE( int sys_trace, (int req, int proc, long addr, long *data_p));
37
38 _PROTOTYPE( int sys_svrctl, (int proc, int req, int priv,vir_bytes argp));
39 _PROTOTYPE( int sys_nice, (int proc, int priority));
40
41 _PROTOTYPE( int sys_int86, (struct reg86u *reg86p));
42
43 /* Shorthands for sys_sdevio() system call. */
44 #define sys_insb(port, proc_nr, buffer, count) \
45         sys_sdevio(DIO_INPUT, port, DIO_BYTE, proc_nr, buffer, count)
46 #define sys_insw(port, proc_nr, buffer, count) \
47         sys_sdevio(DIO_INPUT, port, DIO_WORD, proc_nr, buffer, count)
48 #define sys_outsb(port, proc_nr, buffer, count) \
49         sys_sdevio(DIO_OUTPUT, port, DIO_BYTE, proc_nr, buffer, count)
50 #define sys_outsw(port, proc_nr, buffer, count) \
51         sys_sdevio(DIO_OUTPUT, port, DIO_WORD, proc_nr, buffer, count)
52 _PROTOTYPE( int sys_sdevio, (int req, long port, int type, int proc_nr,
53         void *buffer, int count));
54
55 /* Clock functionality: get system times or (un)schedule an alarm call. */
56 _PROTOTYPE( int sys_times, (int proc_nr, clock_t *ptr));
57 _PROTOTYPE(int sys_setalarm, (clock_t exp_time, int abs_time));
58
59 /* Shorthands for sys_irqctl() system call. */
60 #define sys_irqdisable(hook_id) \
61     sys_irqctl(IRQ_DISABLE, 0, 0, hook_id)
62 #define sys_irqenable(hook_id) \
63     sys_irqctl(IRQ_ENABLE, 0, 0, hook_id)
64 #define sys_irqsetpolicy(irq_vec, policy, hook_id) \
65     sys_irqctl(IRQ_SETPOLICY, irq_vec, policy, hook_id)
66 #define sys_irqrmpolicy(irq_vec, hook_id) \
67     sys_irqctl(IRQ_RMPOLICY, irq_vec, 0, hook_id)
68 _PROTOTYPE ( int sys_irqctl, (int request, int irq_vec, int policy,
69     int *irq_hook_id) );
70
71 /* Shorthands for sys_vircopy() and sys_physcopy() system calls. */
72 #define sys_biosin(bios_vir, dst_vir, bytes) \
73         sys_vircopy(SELF, BIOS_SEG, bios_vir, SELF, D, dst_vir, bytes)
```



```
74 #define sys_biosout(src_vir, bios_vir, bytes) \
75         sys_vircopy(SELF, D, src_vir, SELF, BIOS_SEG, bios_vir, bytes)
76 #define sys_datacopy(src_proc, src_vir, dst_proc, dst_vir, bytes) \
77         sys_vircopy(src_proc, D, src_vir, dst_proc, D, dst_vir, bytes)
78 #define sys_textcopy(src_proc, src_vir, dst_proc, dst_vir, bytes) \
79         sys_vircopy(src_proc, T, src_vir, dst_proc, T, dst_vir, bytes)
80 #define sys_stackcopy(src_proc, src_vir, dst_proc, dst_vir, bytes) \
81         sys_vircopy(src_proc, S, src_vir, dst_proc, S, dst_vir, bytes)
82 _PROTOTYPE(int sys_vircopy, (int src_proc, int src_seg, vir_bytes src_vir,
83         int dst_proc, int dst_seg, vir_bytes dst_vir, phys_bytes bytes));
84
85 #define sys_abscopy(src_phys, dst_phys, bytes) \
86         sys_physcopy(NONE, PHYS_SEG, src_phys, NONE, PHYS_SEG, dst_phys, bytes)
87 _PROTOTYPE(int sys_physcopy, (int src_proc, int src_seg, vir_bytes src_vir,
88         int dst_proc, int dst_seg, vir_bytes dst_vir, phys_bytes bytes));
89 _PROTOTYPE(int sys_memset, (unsigned long pattern,
90                 phys_bytes base, phys_bytes bytes));
91
92 /* Vectored virtual / physical copy calls. */
93 #if DEAD_CODE           /* library part not yet implemented */
94 _PROTOTYPE(int sys_virvcopy, (phys_cp_req *vec_ptr,int vec_size,int *nr_ok));
95 _PROTOTYPE(int sys_physvcopy, (phys_cp_req *vec_ptr,int vec_size,int *nr_ok));
96 #endif
97
98 _PROTOTYPE(int sys_umap, (int proc_nr, int seg, vir_bytes vir_addr,
99          vir_bytes bytes, phys_bytes *phys_addr));
100 _PROTOTYPE(int sys_segctl, (int *index, u16_t *seg, vir_bytes *off,
101         phys_bytes phys, vir_bytes size));
102
103 /* Shorthands for sys_getinfo() system call. */
104 #define sys_getkmessages(dst)   sys_getinfo(GET_KMESSAGES, dst, 0,0,0)
105 #define sys_getkinfo(dst)       sys_getinfo(GET_KINFO, dst, 0,0,0)
106 #define sys_getmachine(dst)     sys_getinfo(GET_MACHINE, dst, 0,0,0)
107 #define sys_getproctab(dst)     sys_getinfo(GET_PROCTAB, dst, 0,0,0)
108 #define sys_getprivtab(dst)     sys_getinfo(GET_PRIVTAB, dst, 0,0,0)
109 #define sys_getproc(dst,nr)     sys_getinfo(GET_PROC, dst, 0,0, nr)
110 #define sys_getrandomness(dst)  sys_getinfo(GET_RANDOMNESS, dst, 0,0,0)
111 #define sys_getimage(dst)       sys_getinfo(GET_IMAGE, dst, 0,0,0)
112 #define sys_getirqhooks(dst)    sys_getinfo(GET_IRQHOOKS, dst, 0,0,0)
113 #define sys_getmonparams(v,vl)  sys_getinfo(GET_MONPARAMS, v,vl, 0,0)
114 #define sys_getschedinfo(v1,v2) sys_getinfo(GET_SCHEDINFO, v1,0, v2,0)
115 #define sys_getlocktimings(dst) sys_getinfo(GET_LOCKTIMING, dst, 0,0,0)
116 #define sys_getbiosbuffer(virp, sizep) sys_getinfo(GET_BIOSBUFFER, virp, \
117         sizeof(*virp), sizep, sizeof(*sizep))
118 _PROTOTYPE(int sys_getinfo, (int request, void *val_ptr, int val_len,
119                              void *val_ptr2, int val_len2)          );
120
121 /* Signal control. */
122 _PROTOTYPE(int sys_kill, (int proc, int sig) );
123 _PROTOTYPE(int sys_sigsend, (int proc_nr, struct sigmsg *sig_ctxt) );
124 _PROTOTYPE(int sys_sigreturn, (int proc_nr, struct sigmsg *sig_ctxt) );
125 _PROTOTYPE(int sys_getksig, (int *k_proc_nr, sigset_t *k_sig_map) );
126 _PROTOTYPE(int sys_endksig, (int proc_nr) );
127
128 /* NOTE: two different approaches were used to distinguish the device I/O
129  * types 'byte', 'word', 'long': the latter uses #define and results in a
130  * smaller implementation, but looses the static type checking.
131  */
132 _PROTOTYPE(int sys_voutb, (pvb_pair_t *pvb_pairs, int nr_ports)         );
133 _PROTOTYPE(int sys_voutw, (pvw_pair_t *pvw_pairs, int nr_ports)         );
134 _PROTOTYPE(int sys_voutl, (pvl_pair_t *pvl_pairs, int nr_ports)         );
135 _PROTOTYPE(int sys_vinb, (pvb_pair_t *pvb_pairs, int nr_ports)          );
136 _PROTOTYPE(int sys_vinw, (pvw_pair_t *pvw_pairs, int nr_ports)          );
137 _PROTOTYPE(int sys_vinl, (pvl_pair_t *pvl_pairs, int nr_ports)          );
138
139 /* Shorthands for sys_out() system call. */
140 #define sys_outb(p,v)   sys_out((p), (unsigned long) (v), DIO_BYTE)
141 #define sys_outw(p,v)   sys_out((p), (unsigned long) (v), DIO_WORD)
142 #define sys_outl(p,v)   sys_out((p), (unsigned long) (v), DIO_LONG)
143 _PROTOTYPE(int sys_out, (int port, unsigned long value, int type)       );
144
145 /* Shorthands for sys_in() system call. */
146 #define sys_inb(p,v)    sys_in((p), (unsigned long*) (v), DIO_BYTE)
```

```
147 #define sys_inw(p,v)    sys_in((p), (unsigned long*) (v), DIO_WORD)
148 #define sys_inl(p,v)    sys_in((p), (unsigned long*) (v), DIO_LONG)
149 _PROTOTYPE(int sys_in, (int port, unsigned long *value, int type)      );
150
151 #endif /* _SYSLIB_H */
152
```