UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION

INTERNATIONAL CENTRE FOR SCIENCE AND HIGH TECHNOLOGY
c/o INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS 34100 TRIESTE (ITALY) VIA GRIGNANO, 9 (ADRIATICO PALACE) P.O. BOX 586 TELEPHONE 040-224572 TELEFAX 040-224575 TELEX 460449 APH I

SMR/643 - 5

SECOND COLLEGE ON
MICROPROCESSOR-BASED REAL-TIME CONTROL -
PRINCIPLES AND APPLICATIONS IN PHYSICS
5 - 30 October 1992

ADDITION TO INTRODUCTION TO REAL-TIME OPERATING SYSTEM

C. VERKERK
Computing and Networks Division
CERN
Geneva
Switzerland

SYSTEM CALLS

FROM AN ASSEMBLY LANGUAGE PROGRAM:

- LOAD VALUES INTO REGISTERS
- osg F$some (or I$thing)
- bcs Error

EXAMPLE:

```
        lda   <path     path (in direct page) # pn.
        ldx   <bufadr   bufadr contains address
        ldy   #80       read 80 characters max.
        osg   I$ReadLn  read the line
        bcs   Error
        sty   <len      len will contain number of
                          characters actually read
        :
Normal  clrb
Error   rts
```

NOTE: THE PATH HAD BEEN OPENED PREVIOUSLY

FROM A C PROGRAM:

```
#include <stdio.h>
main()
int k,size, number;
FILE *fp;
char buffer[80];
size=1;
number=80;
fp = fopen("myfile","r");
        :
k=fread(buffer, size, number, fp);
```

clib.l (/rø/LIB/clib.l)

CONTAINS MANY THINGS!

1. "STANDARD" C LIBRARY:
   - HIGH-LEVEL I/O FUNCTIONS
   - ARITHMETIC, STRING HANDLING FUNCTIONS
2. C SYSTEM CALLS
   - (PSEUDO) EQUIVALENTS OF UNIX CALLS
   - LOW-LEVEL I/O CALLS
   - OSq-SPECIFIC CALLS

STANDARD C LIBRARY:

HIGH-LEVEL I/O USES A POINTER TO A STRUCTURE FILE, DEFINED IN stdio.h
DON'T CONFUSE WITH "PATH NUMBER", USED BY OSq (PATH NUMBER IS 1 BYTE, OR IN A C PROGRAM, AN INTEGER)

IMPORTANT HIGH-LEVEL I/O FUNCTIONS:

```
#include <stdio.h>
main()
FILE *fp;
filename = "myfile";
char buffer[256];          (or buffer[BUFSIZ])
char *action;
int size, number, len, place, n
action = "r";              ("a" for append; "w" for write,
fp = fopen(filename, action);
len = fread(buffer, size, number, fp);
fclose(fp);
```

fwrite

MORE HIGH-LEVEL I/O FUNCTIONS:

```
fflush(fp)
feof(fp), ferror(fp), clearerr(fp), fileno(fp)
fseek(fp, offset, place)        long offset;
rewind(fp)
long ftell(fp)
getc(fp), getchar(), getw(fp)
char *gets(s)                   char *s;
char *fgets(s, n, fp)           char *s;
fprintf(fp, control [,arg0[,arg1 ....]])    char *contro
sprintf(string, control [,arg0[,arg1 ...]])  char *string
putc(ch, fp)                    char ch;
putchar(ch), putw(n, fp)
puts(s), fputs(s, fp)           char *s;
fscanf(fp, control [, pointer])
sscanf(buffer, control [, pointer .....])
setbuf(fp, buffer)
ungetc(ch, fp)
```

UTILITY FUNCTIONS:

- CONVERSION ASCII ⟷ NUMBER (int, long, float
- STRING HANDLING (COMPARE, COPY, CONCATENATE
- STRING SEARCH
- CHARACTER CLASSIFICATION (ISALPHA, ISDIGIT
- SORTING
- MEMORY ALLOCATION:

```
CHAR *MALLOC(SIZE)      unsigned size;
free(ptr)               char *ptr;
char *calloc(nel, elsize)   unsigned nel, elsize;
```

IMPORTANT UTILITY FUNCTIONS:

system(string)	char *string
sleep(seconds)	unsigned seconds
longjmp(env, val)	A SORT OF GOTO, but SPANNING MORE THAN ONE LEVEL OF FUNCTIONS.
setjmp(env)	

EXAMPLE:

```
#include <setjmp.h>
jmp_buf pos;

main()
int .....
{
intercept(trap);          /* SET UP TO RECEIVE SIGNAL */
        DO THINGS
setjmp(pos);              /* DEFINE POSITION FOR longjmp */
    DO MORE THINGS        (IN AN INFINITE LOOP, FOR INSTANCE)
exit(0);                  ← NOT THERE IF INFINITE LOOP
}

trap(sig)
{ TAKE ACTION REQUIRED BY SIGNAL
longjmp(pos, 1);
}
```

OSg System Calls from C.

Majority corresponds directly to OSg system calls. Name may be different for reasons of compatibility with other systems.

I/O functions:

ATTENTION! `creat(fname, perm)` — `*char fname;`
`access(fname, perm)` — `int perm;`
ATTENTION! `open(fname, mode)` — `int mode;`
`close(pn)` — `int pn;`
`dup(pn)`
`read(pn, buffer, count)` — `char *buffer;`
`readln(pn, buffer, count)`
`write(pn, buffer, count)`
`writeln(pn, buffer, count)`
`long lseek(pn, position, type)` — `long position;`
`getstat(---, --, ---)` } comb in different flavours for standard RBF and SCF devices.
`setstat(---, ---, ---)` }

For non-standard devices, getstat and setstat must be coded in the device driver.

- use `#include <modes.h>` for creat, access, open
- it may be convenient to use
  `#include <osg.h>`
  for several calls.

EXAMPLE:

```
#include <osg.h>
#include <modes.h>
/* DOES AN I$GETSTT TO GET FILESIZE */

main(argc, argv)
int argc;
char **argv;
{
    struct registers reg;
    int path;
/* FOR THE LINKER: */
    pflinit();
    path = open(*++argv, S_IREAD);
    reg.rg_a = path;
    reg.rg_b = SS_SIZE;

    if((_osg(I_GETSTT, &reg) == 0)
        printf("filesize = %lx\n",
            (long)(reg.rg_x << 16)+reg.rg_u);
    else printf("OSg error # %d\n", reg.rg_b & 0xff);
    dumpregs(&reg);
}

dumpregs(r)
register struct registers *r;
{
printf("cc=%02x\n", r->rg_cc & 0xff);
printf("x = %04x\n", r->rg_x);
}
```

OF PARTICULAR IMPORTANCE TO US ARE:

LOADING, LINKING, ETC.:

```
#include <module.h>
mod_exec *modlink(modname, type, language)
mod_exec *modload(filename, type, language)
char *modname, *filename;
munlink(mod)
mod_exec *mod;
```

(modload, modlink return a pointer to the load address of the module; munlink needs this pointer as argument)

```
osgfork(modname, parsize, parptr, type, lang, datasize)
  char *modname, parptr;
```

osgfork returns PROCESS ID of child process.

SIGNAL HANDLING:

```
kill(tid, signal)
```

(SENDS "signal" TO TASK "tid")

```
intercept(func)      /* "func" is a pointer to a */
   int (*func)()     /* function returning an integer */

(*signal(intrupt, address))()
(*address)();
```

(i.e. "SIGNAL" RETURNS A POINTER TO A FUNCTION
"ADDRESS" IS A POINTER TO A FUNCTION)

("signal" is equivalent to "intercept + switch")

→ "signal" is incompatible with "intercept"!

EXAMPLE:

In main body:

```
pn = creat("temp",3);
signal(2, trap);
signal(3, trap);
write(pn, string, count);
close(pn);
unlink("temp");
exit(0);

trap(sig)
{
close(pn);
unlink("temp");
exit(sig);
}
```

DON'T FORGET: #include <signal.h>

SIGNALS AND SYNCHRONISATION:

```
pause()          /* HALTS TASK UNTILL SIGNAL IS RECEIVED
wait(status)     /* WAITS FOR TERMINATION OF TASK
   int *status;        (CHILD) */
tsleep(ticks)    /* DEACTIVATES THE CALLER */

#include <time.h>
setime(buffer)
getime(buffer)
struct satbuf *buffer   /* defined in time.h */
```

DO-ALL SYSTEM CALL:

```
# include <osg.h>
_osg(code, reg)
char code;
struct registers *reg;
```

IN osg.h struct registers IS DEFINED:

```
struct registers {
    char rg_cc, rg_a, rg_b, rg_dp;
    unsigned rg_x, rg_y, rg_u;
    };
```