Subsections

C, UNIX and Standard Libraries

There is a very close link between C and most operating systems that run our C programs. Almost the whole of the UNIX operating system is written in C. This Chapter will look at how C and UNIX interface together.

We have to use UNIX to maintain our file space, edit, compile and run programs ***etc.***.

However UNIX is much more useful than this:

Advantages of using UNIX with C

- **Portability** -- UNIX, or a variety of UNIX, is available on many machines. Programs written in ***standard*** UNIX and C should run on any of them with little difficulty.
- **Multiuser / Multitasking** -- many programs can share a machines processing power.
- **File handling** -- hierarchical file system with many file handling routines.
- **Shell Programming** -- UNIX provides a powerful command interpreter that understands over 200 commands and can also run UNIX and user-defined programs.

- **Pipe** -- where the output of one program can be made the input of another. This can done from command line or within a C program.
- **UNIX utilities** -- there over 200 utilities that let you accomplish many routines without writing new programs. ***e.g.*** make, grep, diff, awk, more ....
- **System calls** -- UNIX has about 60 system calls that are at the ***heart*** of the operating system or the ***kernel*** of UNIX. The calls are actually written in C. All of them can be accessed from C programs. Basic I/0, system clock access are examples. The function `open()` is an example of a system call.
- **Library functions** -- additions to the operating system.

Using UNIX System Calls and Library Functions

To use system calls and library functions in a C program we simply call the appropriate C function.

Examples of standard library functions we have met include the higher level I/O functions -- `fprintf()`, `malloc()` ...

Aritmetic operators, random number generators -- `random()`, `srandom()`, `lrand48()`, `drand48()` ***etc.*** and basic C types to string conversion are memebers of the `stdlib.h` standard library.

All math functions such as `sin()`, `cos()`, `sqrt()` are standard math library (`math.h`) functions and others follow in a similar fashion.

For most system calls and library functions we have to include an appropriate header file. ***e.g.*** `stdio.h, math.h`

To use a function, ensure that you have made the required `#includes` in your C file. Then the function can be called as though you had defined it yourself.

It is important to ensure that your arguments have the expected types, otherwise the function will probably produce strange results. `lint` is quite good at checking such things.

Some libraries require extra options before the compiler can support their use. For example, to compile a program including functions from the `math.h` library the command might be

```
 cc mathprog.c -o mathprog -lm
```

The final `-lm` is an instruction to link the maths library with the program. The manual page for each function will usually inform you if any special compiler flags are required.

Information on nearly all system calls and library functions is available in manual pages. These are available on line: Simply type `man` function name.

e.g. `man drand48`

would give information about this random number generator.

Over the coming chapters we will be investigating in detail many aspects of the C Standard Library and also other UNIX libraries.

Dave Marshall
1/5/1999