CSC / CPE 357

Systems Programming

Topics

- I/O
- Strings
- C library functions

These slides are based on materials from Professors Aaron Keen and Andrew Migler

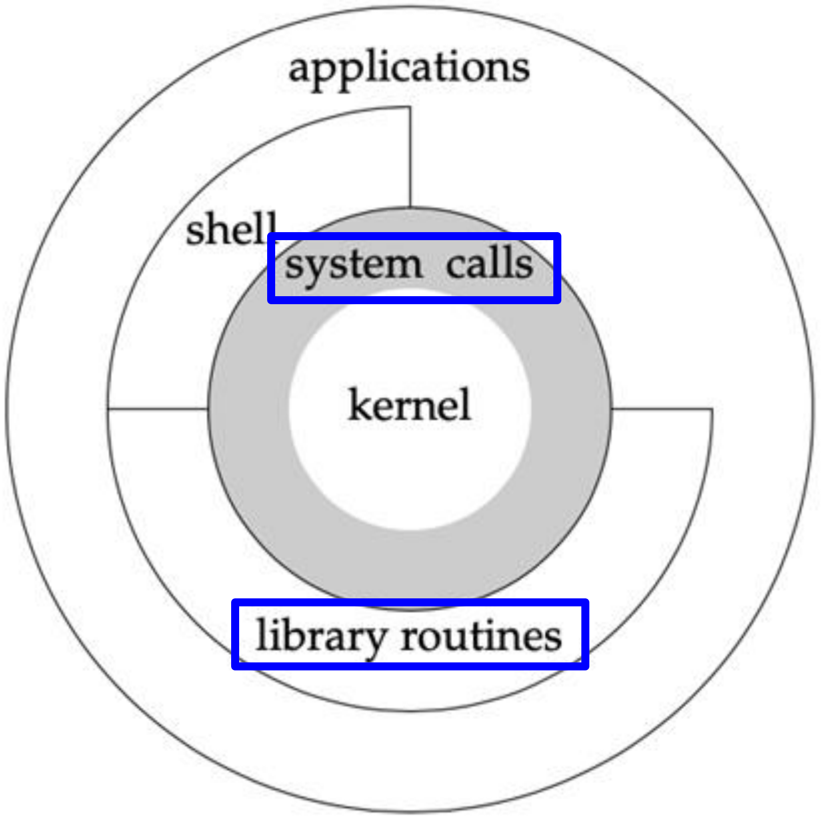


Figure 1.1 Architecture of the UNIX operating system

Diagram from: Advanced Programming in the UNIX Environment, 3rd Ed.

System Calls and Library Functions

- **System call**: entry point directly into the kernel
  - Linux provides ~400 system calls
  - Exposed as regular C functions

- Contrast with: **library functions**, which do not represent a direct entry point into the kernel
  - `printf()` library function invokes the `write()` system call
  - Many library functions do not involve system calls, examples:
    - `strcpy()` copy a string
    - `atoi()` convert ASCII to integer

- Manual pages:
  - "section 2" for system calls: `man 2 read`
  - "section 3" for library functions: `man 3 getchar`

System Calls and Library Functions

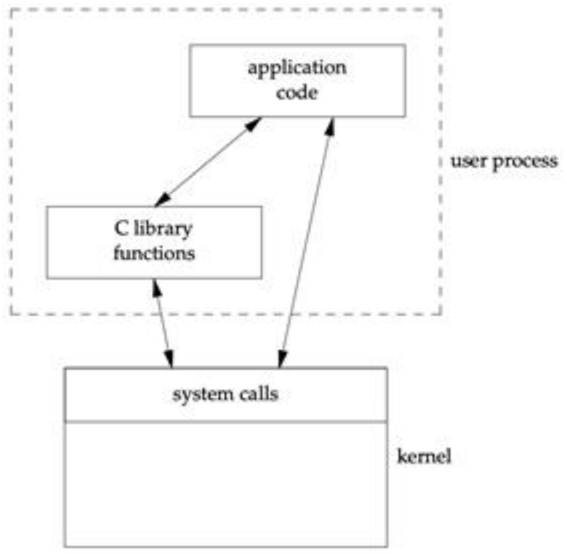


Figure 1.12 Difference between C library functions and system calls

- An application can make a system call directly or call a library routine.
- System calls usually provide a minimal interface, library functions often provide more functionality.

Example Library Functions

- atoi() - convert a string to an integer
- strcpy() - copy a string
- strcmp() - compare two strings
- strlen() - calculate the length of a string

Example System Calls

- `open()` - open and possibly create a file
- `read()` - read from a file descriptor
- `time()` - returns the time as the number of seconds since 1970-01-01
- `socket()` - create an endpoint for communication

Input/Output

- Unbuffered I/O
    - Each read or write invokes a system call in the kernel
    - Part of the UNIX Specification
    - `open()`
    - `close()`
    - `read()`
    - `write()`
    - `lseek()`

(these are all system calls)

- Standard I/O
    - Part of the C Standard Library (`stdio.h`)
    - Buffering provided to minimize the number of `read()` and `write()` system calls

Standard I/O

- The standard I/O library, defined in `<stdio.h>` implements a model of input/output based on **streams**

- Text stream: sequence of lines, each line ends with a newline `\n`

- Simplest input/output mechanism: read/write one character at a time:
  - `int getchar(void)` - returns next character from `stdin`, or `EOF` (End of File)
  - `int putchar(int)` - returns character written or `EOF` in case of error

Standard I/O: Formatted Output

- We have used `printf()` informally.
  - `int printf(char *format, arg1, arg2, …);`

- The `format` string may contain:
  - Regular characters
  - Conversion specifications, beginning with `%`:
    - `%d`, `%i` - integer
    - `%o` - unsigned octal number
    - `%x` - unsigned hexadecimal number
    - `%c` - single character
    - `%s` - string (char *)
    - `%f` - double
  - Conversion specification may include formatting options (width, padding) one example:
    - `%.2f` (floating point, with precision 2)

Standard I/O: Formatted Input

```
int scanf(char *format, …)
```

`scanf` reads characters from standard input, interpreting them according to `format`

Important: arguments to `scanf()` must be pointers. We will revisit this topic soon.

A Few Standard Library Functions

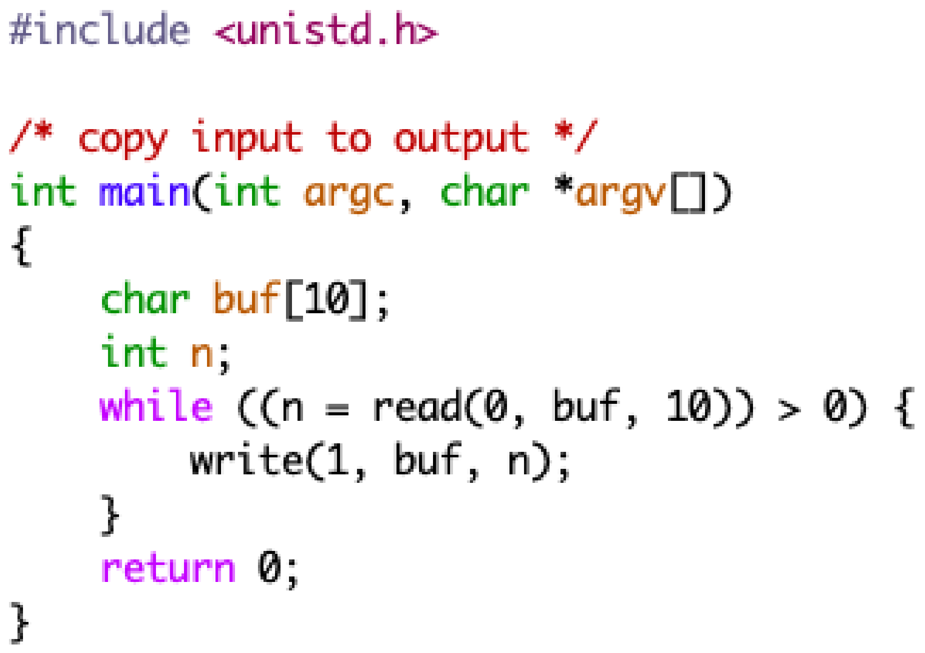
- `<string.h>`
  - `strcat()` - concatenate strings
  - `strcmp()` - compare strings

- `<ctype.h>`
  - `isalpha()`
  - `isupper()`
  - `islower()`
  - `isdigit()`
  - `isspace()`

A Few Core Development Tools

- `gcc`: GNU C Compiler
- `gdb`: Debugger used with C programs. Requires `gcc -g`
- `time`: Measures how long a program takes to run.
- `valgrind`: Programming tool for memory debugging, memory leak detection, and profiling.

Example: Copy Input to Output (System Calls)



```c
#include <unistd.h>

/* copy input to output */
int main(int argc, char *argv[])
{
    char buf[10];
    int n;
    while ((n = read(0, buf, 10)) > 0) {
        write(1, buf, n);
    }
    return 0;
}
```

Gist