Embedded Linux Programming

Mahesh U. Patil

Version 0.1, 2016

NOTE Some examples are specific to Beaglebone black.

Unix Structure

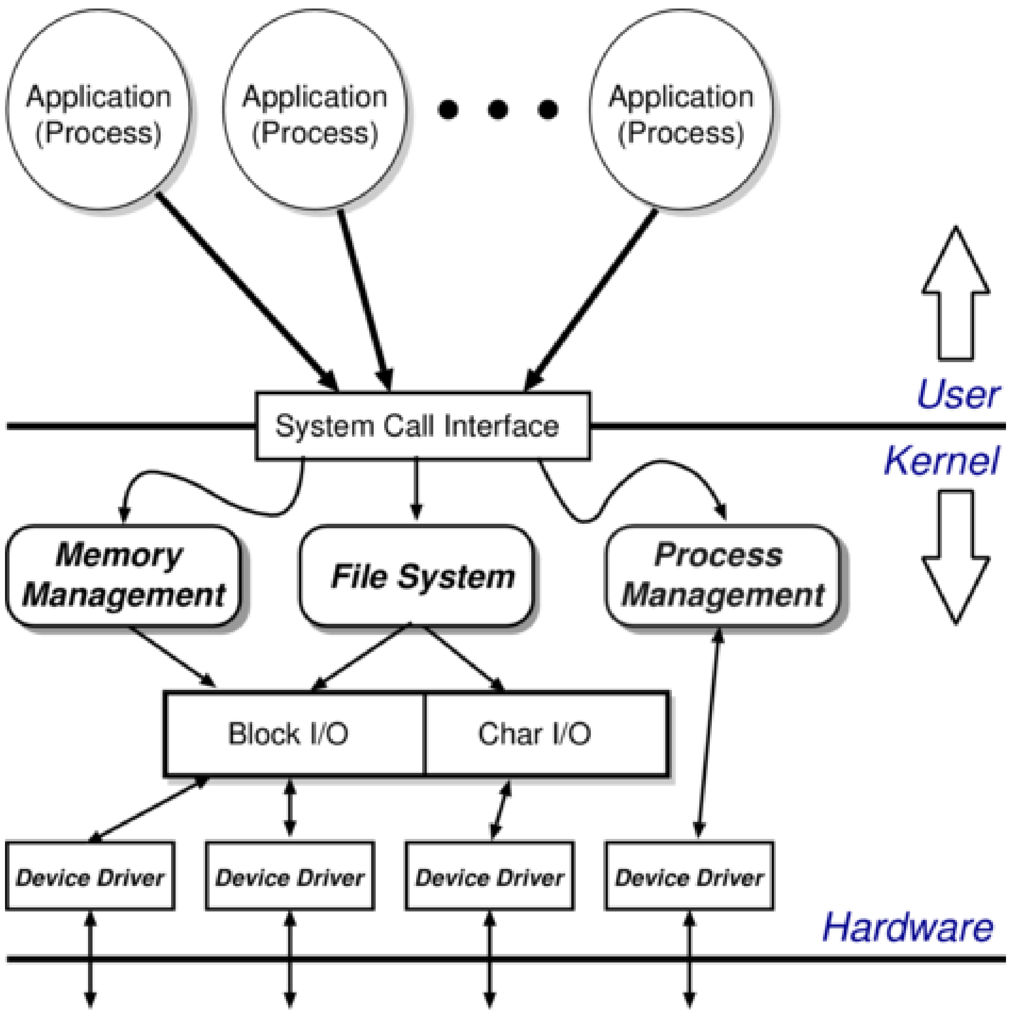

Figure 1. Unix Structure

Build Process

1. pre-processor, compiler, linker, assembler, debugger, object utilities
2. Makefile

[comment] Write a makefile that compiles code for x86 and ARM platform. Also add command to mafefile to transfer the source code and binary to the target board.

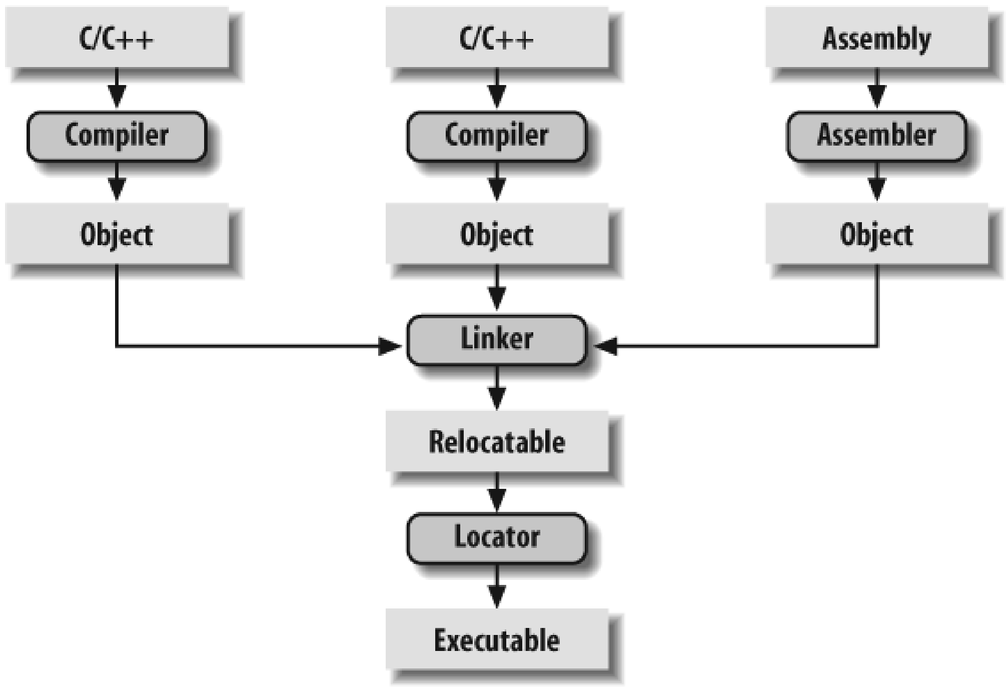

Figure 2. Build process

Understanding ELF format

1. Linking and Execution view
2. Segments vs Sections: The segments contain information that is necessary for runtime execution of the file, while sections contain important data for linking and relocation
3. Four main sections: .text, .data, .rodata, and .bss
4. Code, Data, BSS, Heap, Stack (Stack Frame)
5. Entry point for a program
    a. readelf --headers ./main | grep "Entry point"
    b. objdump --disassemble ./main | grep "address from above readelf command"
    c. C Run Time (CRT)

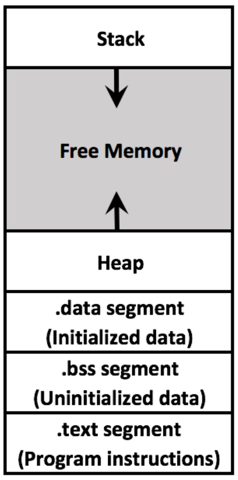
Stack
Free Memory
Heap
.data segment (Initialized data)
.bss segment (Uninitialized data)
.text segment (Program instructions)

Figure 3. ELF

General Topics

1. extern int errno, errno.h
2. perror, strerror, strerror_r (thread safe)
3. exit, stdlib.h, EXIT_SUCCESS, EXIT_FAILURE
4. API vs ABI
5. Blocking and Non-Blocking API Calls
6. Buffered, Non-Buffered, Formatted, Non-Formatted inputs/ outputs
7. Opaque data types

NOTE Write a program to demonstrate the behavior of buffered and formatted I/O.

Library Calls vs System Calls

TIP read man pages for ltrace and strace

Library Calls

1. Static Library
2. Dynamic Library

NOTE Write a program to develop a arithmetic static (libarith.a) and dynamic (libarith.so) library.

System Calls

1. open, read, write, close System Calls
2. open

   ```
   int open(const char *pathname, int flags, mode_t mode);
   ```

   a. flags, O_RDONLY, O_WRONLY, or O_RDWR
   b. flags, O_APPEND, O_CREAT, O_NONBLOCK (does not work on regular files), O_TRUNC
   c. mode, S_IRWXU, S_IRUSR, S_IRWXG, S_IRGRP, S_IRWXO, S_IROTH
3. read

   ```
   ssize_t read(int fd, void *buf, size_t count);
   ```

   i. 0 → EOF,
   ii. Non-blocking read

   ```
   char buf[BUFSIZ];
   ssize_t nr;
   start:
     nr = read (fd, buf, BUFSIZ);
     if (nr == -1) {
       if (errno == EINTR)
         goto start;
       if (errno == EAGAIN)
         /* resubmit later */
       else
         /* error */
     }
   ```

4. write

```
ssize_t write (int fd, const void *buf, size_t count);
```

   a. partial write example

   ```
   ssize_t ret, nr;
   while (len != 0 && (ret = write (fd, buf, len)) != 0) {
     if (ret == -1) {
       if (errno == EINTR)
         continue;
       perror ("write");
     break;
     }
     len -= ret;
     buf += ret;
   }
   ```

5. close

```
int close (int fd);
```

NOTE Write a program using system calls to replicate the behavior of copy command "cp"

1. fsync

```
int fsync (int fd); // Flushes both data and metadata (timestamps, other
attributes..)
int fsyncdata (int fd); // Flushes on data, not guarantee of metadata
```

   a. O_SYNC flag in open can also be used to force synchronization
   b. O_DSYNC - Defined by POSIX, same as fsyncdata()
   c. O_RSYNC - Defined by POSIX, same as fsync()

2. lseek

```
#include<unistd.h>
#include<sys/types.h>
off_t lseek (int fd, off_t pos, int origin);
```

   a. SEEK_CUR The current file position of fd is set to its current value plus pos, which can be negative, zero, or positive. A pos of zero returns the current file position value.
   b. SEEK_END The current file position of fd is set to the current length of the file plus pos,

   which can be negative, zero, or positive. A pos of zero sets the offset to the end of the file.

   c. SEEK_SET The current file position of fd is set to pos. A pos of zero sets the offset to the beginning of the file.

3. pipes, O_NONBLOCK, fcntl(fd, F_SETFL, fcntl(fd, F_GETFL) | O_NONBLOCK);
4. Schedulers (Short, Long and Medium term)

Important opaque data types

```
size_t
ssize_t
pid_t
off_t
```

References

[book] Books

1. Linux System Programming: Talking Directly to the Kerel and C Library, 2nd Edition, Robert Love
2. Programming Embedded Systems, 2nd Edition, Michael Barr