Unix Programming

Unix File System

Nguyen Thanh Hung
Software Engineering Department
Hanoi University of Science and Technology

Outline

- What is File System?
- Important Directories in Linux
- Mounting File System
- Useful commands and tools
- Programming with Files

What is File System

- It is responsible for storing information on disk and retrieving and updating this information.
- Example :
  - FAT16, FAT32, NTFS
  - ext2, ext3
  - …
- In Linux everything is file.

Type of File System

- **Network File System**
  - NFS
  - SMB (Server Message Block)
- **Disk File System**
  - ext2
  - ext3
  - FAT32
  - NTFS (New Technology File System)

Network File System

- Network File System are physically somewhere else, but appear as if they are mounted on one computer.
- NFS
  - It was developed by Sun.
- SMB
  - It was developed by Microsoft.

Disk File System

- Disk File System are what you will find on a physical device, such as hard drive in a computer.

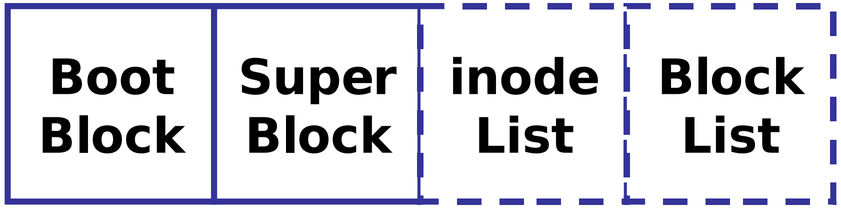
ext2 File System

- It has been the standard File System for Linux.
- The original Extended File System was named ext.
- The ext2 File System can accommodate:
  - Files as large as 2GB
  - Directories as large as 2TB
  - Max. file name length of 255 characters.

ext2 Structure

- A file in the ext2 File System begins with the inode.
- inode
  - Each file has an inode structure that is identified by an i-number.
  - The inode contains the information required to access the file.
  - It doesn’t contain file name.

Physical Structure on the Disk



Boot Block | Super Block | inode List | Block List

- Boot Block : information needs to boot the system
- Super Block : File System Specifications
  - Size
  - Max. number of files
  - Free blocks
  - Free inodes
- inode List
- Block List : The files data

Symbolic Link

- Because of the structure of the ex2 File System, several names can be associated with a single file.
- In effect, you create another inode that reference already existing data.

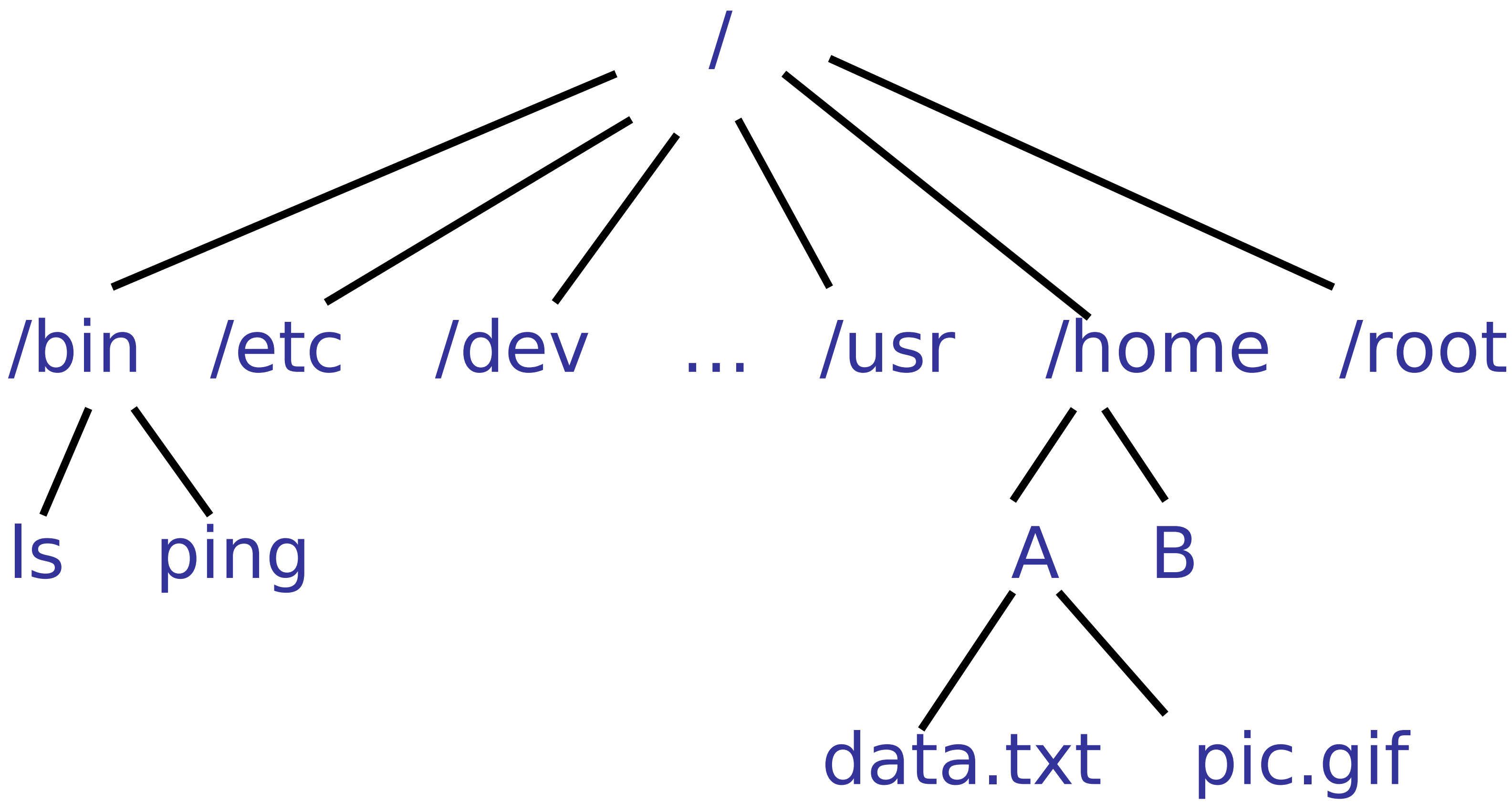
ext3 File System

- It is as same as ext2.
- It is a journaling File System for Linux.
- In a journaling system, metadata is written to a journal on the disk before it is actually used to modify the file.

Outline

- What is File System?
- **Important Directories in Linux**
- Mounting File System
- Useful commands and tools
- Programming with Files

File System Structure



/bin

- Hold the most commonly used essential user programs
  - login
  - Shells (bash, ksh, csh)
  - File manipulation utilities (cp, mv, rm, ln, tar)
  - Editors (ed, vi)
  - File system utilities (dd, df, mount, umount, sync)
  - System utilities (uname, hostname, arch) GNU utilities like gzip and gunzip

/sbin

- Hold essential maintenance or system programs such as the following:
  - fsck
  - Fdisk
  - Mkfs
  - Shutdown
  - Lilo
  - Init
  - …
- The main difference between the programs stored in /bin and /sbin is that the programs in /sbin are executable only by root.

/etc

- Store the system wide configuration files required by many programs.
  - passwd
  - shadow
  - fstab
  - hosts
  - lilo.conf
  - ...

/home and /root

- The /home directory is where all the home directories for all the users on a system are stored.
- The /root directory is where all the home directories for root user on a system are stored.

/dev

- The special files representing hardware are kept in it.
  - /dev/hda1
  - /dev/ttyS0
  - /dev/mouse
  - /dev/fd0
  - /dev/fifo1
  - /dev/loop2
  - …

/tmp and /var

- The /tmp and /var directories are used to hold temporary files or files with constantly varying content.
- The /tmp directory is usually a dumping ground for files that only need to be used briefly and can afford to be deleted at any time.
- The /var directory is a bit more structured than /tmp and usually looks something like the following:
  - /var/log
  - /var/spool
  - /var/named
  - …

/usr

- Most programs and files directly relating to users of the system are stored.
- It is in some ways a mini version of the / directory.
  - /usr/bin
  - /usr/sbin
  - /usr/spool
  - …

Other directories

- **/mnt**
  - removable media such as CD-ROM, floppy and ... are mounted.
  - /mnt/floppy
  - /mnt/cdrom
- **/boot**
  - Image to boot system
- **/lost+found**
  - Used by fsck

/proc

- It is a virtual File System
- A special File System provided by the kernel as a way of providing information about the system to user programs.
- The main tasks of proc File System is to provide information about the kernel and processes.

Outline

- What is File System?
- Important Directories in Linux
- **Mounting File System**
- Useful commands and tools
- Programming with Files

Mounting File System

- The Linux File System makes it appear as if all the File System are local and mounted somewhere on the root File System.
- File System are mounted with the mount command.
  - mount –t type source mount_point
- To unmount a File System, the umount command is used.
  - umount /dev/<device name> or mount_point

Mounting Automatically with fstab

- This file lists all the partitions that need to be mounted at boot time and the directory where they need to be mounted.
- Along with that information, you can pass parameters to the mount command.
- /etc/fstab
  - Which devices to be mounted
  - What kinds of File Systems they contain
  - At what point in the File System the mount takes place
  - ...

Partitions

- **Primary-Master**
  - /dev/hda
- **Primary-Slave**
  - /dev/hdb
- **Secondary-Master**
  - /dev/hdc
- **Secondary-Slave**
  - /dev/hdd
- **Swap Partition**
  - Used to implement virtual memory

Outline

- What is File System?
- Important Directories in Linux
- Mounting File System
- **Useful commands and tools**
- Programming with Files

Creating File System

- Once a disk has been partitioned for a specific File System, it is necessary to create a File System on it.
- The first process in the DOS world is known as formatting.
- In the UNIX world is known as creating a File System.

Create File System Commands

- **mkfs or mke2fs**
  - Make a new ext2 File System.
- **mk3fs**
  - Make a new ext3 File System.
- **mkdosfs**
  - Make DOS File System without owning any Microsoft software.

FS Commands and Tools

- **pwd**
  - Where am I?
- **cd**
  - Changes working directory.
- **ls**
  - Shows the contents of current directory
- **cat**
  - Takes all input and outputs it to a file or other source
- **mkdir**
  - Creates a new directory
- **rmdir**
  - Removes empty directory

FS Commands and Tools

- **mv**
  - Moves files
- **cp**
  - Copies files
- **rm**
  - Removes directory
- **gzip and gunzip**
  - To compress and uncompress a file
- **tar**
  - To compress and uncompress a file
- **fsck and e2fsck**
  - Checks and repairs a Linux File System (same as scandisk)

FS Commands and Tools

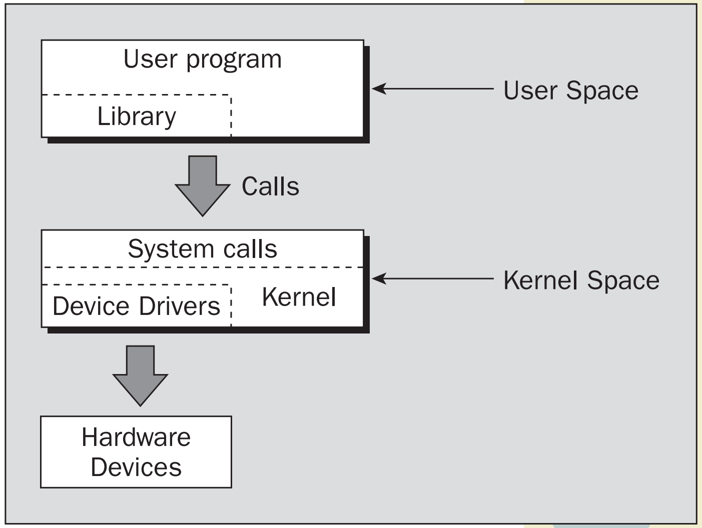
- **e2label**
  - Displays or change the label of a device
- **dd**
  - Converts and copies a file
- **df**
  - Reports File System disk space usage
- **du**
  - Estimates file space usage
- **ln**
  - Makes links between files
- **file**
  - Determines file type

Outline

- What is File System?
- Important Directories in Linux
- Mounting File System
- Useful commands and tools
- **Programming with Files**

System Calls

- The low-level functions used to access the device drivers, the system calls, include:
  - **open**: Open a file or device
  - **read**: Read from an open file or device
  - **write**: Write to a file or device
  - **close**: Close the file or device



File Functions

Write

```
#include <unistd.h>

size_t write(int fildes, const void *buf, size_t nbytes);
```

```
#include <unistd.h>
#include <stdlib.h>

int main()
{
    if ((write(1, "Here is some data\n", 18)) != 18)
        write(2, "A write error has occurred on file descriptor 1\n",46);

    exit(0);
}
```

```
$ ./simple_write
Here is some data
$
```

0: Standard input

1: Standard output

2: Standard error

File Functions

- **Read**

```
#include <unistd.h>

size_t read(int fildes, void *buf, size_t nbytes);
```

```
#include <unistd.h>
#include <stdlib.h>

int main()
{
    char buffer[128];
    int nread;

    nread = read(0, buffer, 128);
    if (nread == -1)
        write(2, "A read error has occurred\n", 26);

    if ((write(1,buffer,nread)) != nread)
        write(2, "A write error has occurred\n",27);
```

```
$ echo hello there | ./simple_read
hello there
$ ./simple_read < draft1.txt
Files
In this chapter we will be looking at files and directories and how to manipulate
them. We will learn how to create files,$
```

File Functions

❖ Open

```
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

int open(const char *path, int oflags);
int open(const char *path, int oflags, mode_t mode);
```

Mode	Description
`O_RDONLY`	Open for read-only
`O_WRONLY`	Open for write-only
`O_RDWR`	Open for reading and writing

`O_APPEND`: Place written data at the end of the file.

`O_TRUNC`: Set the length of the file to zero, discarding existing contents.

`O_CREAT`: Creates the file, if necessary, with permissions given in `mode`.

`O_EXCL`: Used with `O_CREAT`, ensures that the caller creates the file. The `open` is atomic; that is, it's performed with just one function call. This protects against two programs creating the file at the same time. If the file already exists, `open` will fail.

File Functions

- **Close**

```
#include <unistd.h>

int close(int fildes);
```

A File copy Program

```c
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>

int main()
{
    char c;
    int in, out;

    in = open("file.in", O_RDONLY);
    out = open("file.out", O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
    while(read(in,&c,1) == 1)
        write(out,&c,1);

    exit(0);
}
```

```
$ TIMEFORMAT="" time ./copy_system
4.67user 146.90system 2:32.57elapsed 99%CPU
...
$ ls -ls file.in file.out
1029 -rw-r---r-   1 neil     users     1048576 Sep 17 10:46 file.in
1029 -rw-------   1 neil     users     1048576 Sep 17 10:51 file.out
```

Another File Copy Program

```c
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>

int main()
{
    char block[1024];
    int in, out;
    int nread;

    in = open("file.in", O_RDONLY);
    out = open("file.out", O_WRONLY|O_CREAT, S_IRUSR|S_IWUSR);
    while((nread = read(in,block,sizeof(block))) > 0)
        write(out,block,nread);

    exit(0);
}
```

```
$ rm file.out
$ TIMEFORMAT="" time ./copy_block
0.00user 0.02system 0:00.04elapsed 78%CPU
...
```

The Standard I/O Library

lseed

- sets the read/write pointer of a file descriptor

```
#include <unistd.h>
#include <sys/types.h>

off_t lseek(int fildes, off_t offset, int whence);
```

SEEK_SET: `offset` is an absolute position

SEEK_CUR: `offset` is relative to the current position

SEEK_END: `offset` is relative to the end of the file

The Standard I/O Library

fopen

```
#include <stdio.h>

FILE *fopen(const char *filename, const char *mode);
```

`"r"` or `"rb"`: Open for reading only

`"w"` or `"wb"`: Open for writing, truncate to zero length

`"a"` or `"ab"`: Open for writing, append to end of file

`"r+"` or `"rb+"` or `"r+b"`: Open for update (reading and writing)

`"w+"` or `"wb+"` or `"w+b"`: Open for update, truncate to zero length

`"a+"` or `"ab+"` or `"a+b"`: Open for update, append to end of file

The Standard I/O Library

fread

```
#include <stdio.h>

size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
```

fwrite

```
#include <stdio.h>

size_t fwrite (const void *ptr, size_t size, size_t nitems, FILE *stream);
```

fclose

```
#include <stdio.h>

int fclose(FILE *stream);
```

The Standard I/O Library

fflush

- causes all outstanding data on a file stream to be written immediately

```
#include <stdio.h>

int fflush(FILE *stream);
```

fseed

- Sest the position in the stream for the next read or write on the stream

```
#include <stdio.h>

int fseek(FILE *stream, long int offset, int whence);
```

The Standard I/O Library

- **fgetc, fgets, fputc, fputs**
  - Int fgetc(FILE *stream);
  - Int fputc(int c, FILE *stream);

A Third File Copy Program

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int c;
    FILE *in, *out;

    in = fopen("file.in","r");
    out = fopen("file.out","w");

    while((c = fgetc(in)) != EOF)
        fputc(c,out);

    exit(0);
}
```

```
$ TIMEFORMAT="" time ./copy_stdio
0.06user 0.02system 0:00.11elapsed 81%CPU
```

Exercises

- Write a program to append the content of a file to the end of another file
- Write a program to count the number of words in a text file