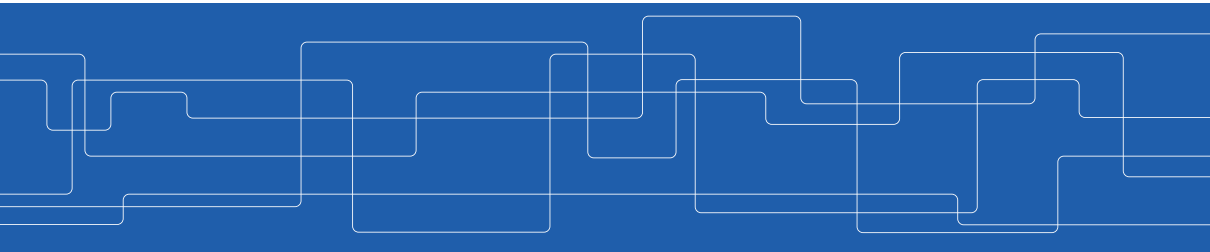
File Systems - Part I

Amir H. Payberah
payberah@kth.se
2022



Motivation

Motivation

- The file system (FS) provides mechanism to access data/programs on storage.

Motivation

- The file system (FS) provides mechanism to access data/programs on storage.
- The FS consists of two distinct parts:

Motivation

- The file system (FS) provides mechanism to access data/programs on storage.
- The FS consists of two distinct parts:
  - A collection of files.

Motivation

- The file system (FS) provides mechanism to access data/programs on storage.

- The FS consists of two distinct parts:
  - A collection of files.
  - A directory structure that organizes and provides information about all the files in the system.

File Concept

File Concept

- Contiguous logical address space.
- Various types.

File Concept

- Contiguous logical address space.
- Various types.

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

File Attributes

- Name: only information kept in human-readable form.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.
- Type: needed for systems that support different types.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.
- Type: needed for systems that support different types.
- Location: pointer to file location on device.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.
- Type: needed for systems that support different types.
- Location: pointer to file location on device.
- Size: current file size.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.
- Type: needed for systems that support different types.
- Location: pointer to file location on device.
- Size: current file size.
- Protection: controls who can do reading, writing, executing.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.
- Type: needed for systems that support different types.
- Location: pointer to file location on device.
- Size: current file size.
- Protection: controls who can do reading, writing, executing.
- Time, date, and user identification: data for protection, security, and usage monitoring.

File Attributes

- Name: only information kept in human-readable form.
- Identifier: unique number identifies file within file system.
- Type: needed for systems that support different types.
- Location: pointer to file location on device.
- Size: current file size.
- Protection: controls who can do reading, writing, executing.
- Time, date, and user identification: data for protection, security, and usage monitoring.
- Information about files are kept in the directory structure.

File Operations

- Create

File Operations

- Create
- Write

File Operations

- Create
- Write
- Read

File Operations

- Create
- Write
- Read
- Delete

File Operations

- Create
- Write
- Read
- Delete
- Open(f): move the content of entry f from disk to memory.

File Operations

- Create
- Write
- Read
- Delete
- Open(f): move the content of entry f from disk to memory.
- Close(f): move the content of entry f in memory to directory structure on disk.

File Locks

- File locks allow one process to lock a file and prevent other processes from gaining access to it.

File Locks

- File locks allow one process to lock a file and prevent other processes from gaining access to it.
- Similar to reader-writer locks.

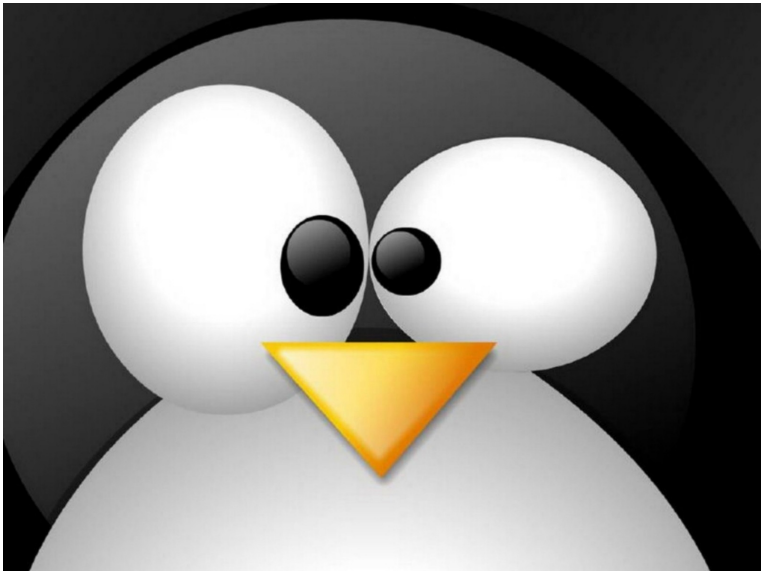
File Locks

- File locks allow one process to lock a file and prevent other processes from gaining access to it.
- Similar to reader-writer locks.
  - Shared lock similar to reader lock: several processes can acquire concurrently

File Locks

- File locks allow one process to lock a file and prevent other processes from gaining access to it.

- Similar to reader-writer locks.
  - Shared lock similar to reader lock: several processes can acquire concurrently
  - Exclusive lock similar to writer lock: only one process can acquire it



Files and Their Metadata

- The `stat` structure: the metadata of a file.
- Defined in `<sys/stat.h>`.

```
struct stat {
  dev_t st_dev;         /* ID of device containing file */
  ino_t st_ino;         /* inode number */
  mode_t st_mode;       /* permissions */
  nlink_t st_nlink;     /* number of hard links */
  uid_t st_uid;         /* user ID of owner */
  gid_t st_gid;         /* group ID of owner */
  dev_t st_rdev;        /* device ID (if special file) */
  off_t st_size;        /* total size in bytes */
  blksize_t st_blksize; /* blocksize for filesystem I/O */
  blkcnt_t st_blocks;   /* number of blocks allocated */
  time_t st_atime;      /* last access time */
  time_t st_mtime;      /* last modification time */
  time_t st_ctime;      /* last status change time */
};
```

Open a File

- `open()` maps the file to a file descriptor.

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

int open(const char *name, int flags);
```

Read and Write

- `read()` and `write()` to read and write from/to a file.

```c
#include <unistd.h>

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

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

The Stat Family

- `stat()` returns information about the file denoted by the path `path`.
- `fstat()` returns information about the file represented by the file descriptor `fd`.

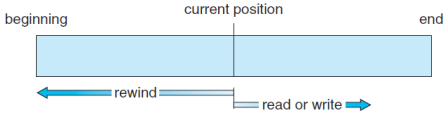
```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *path, struct stat *buf);
int fstat(int fd, struct stat *buf);
```

Access Methods

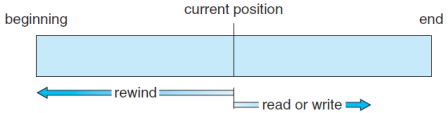
Access Methods - Sequential Access

- Sequential access is based on a tape model of a file.



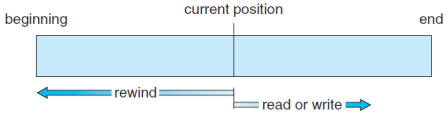
Access Methods - Sequential Access

- Sequential access is based on a tape model of a file.
- Information in the file is processed in order, one record after the other.



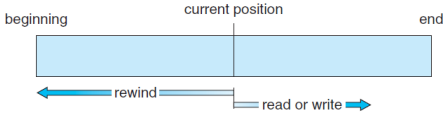
Access Methods - Sequential Access

- Sequential access is based on a tape model of a file.
- Information in the file is processed in order, one record after the other.
- A read operation (`read_next()`): reads the next portion of the file and automatically advances a file pointer.



Access Methods - Sequential Access

- Sequential access is based on a tape model of a file.
- Information in the file is processed in order, one record after the other.
- A read operation (`read_next()`): reads the next portion of the file and automatically advances a file pointer.
- A write operation (`write_next()`): appends to the end of the file and advances to the end of the newly written material.



Access Methods - Direct Access

- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.

Access Methods - Direct Access

- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- Immediate access to large amounts of information.
  - Databases are often of this type.

Access Methods - Direct Access

- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- Immediate access to large amounts of information.
  - Databases are often of this type.
- `read(n)` rather than `read_next()`.
  - `n` is the block number.

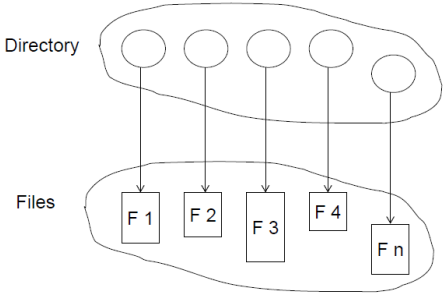
Access Methods - Direct Access

- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- Immediate access to large amounts of information.
  - Databases are often of this type.
- `read(n)` rather than `read_next()`.
  - `n` is the block number.
- `write(n)` rather than `write_next()`.

Directory Structure

Directory Structure

- The directory can be viewed as a symbol table that translates file names into their directory entries.
- Both the directory structure and the files reside on disk.



Operations Performed on Directory

- Search for a file

Operations Performed on Directory

- Search for a file
- Create a file

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file

Operations Performed on Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Directory Organization

- The directory itself can be organized in many ways.

Directory Organization

- The directory itself can be organized in many ways.
  - Single-level directories

Directory Organization

- The directory itself can be organized in many ways.
  - Single-level directories
  - Two-level directories

Directory Organization

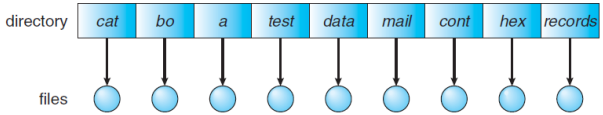
- The directory itself can be organized in many ways.
  - Single-level directories
  - Two-level directories
  - Tree-level directories

Directory Organization

- The directory itself can be organized in many ways.
  - Single-level directories
  - Two-level directories
  - Tree-level directories
  - Acyclic-graph directories

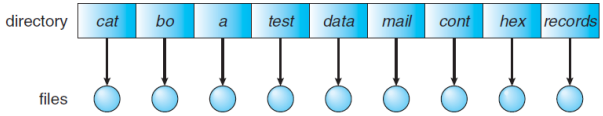
Single-Level Directory

- A single directory for all users.



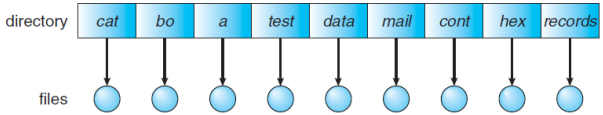
Single-Level Directory

- A single directory for all users.
- Naming problem: they must have unique names.



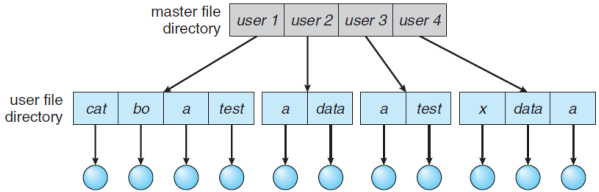
Single-Level Directory

- A single directory for all users.
- Naming problem: they must have unique names.
- Grouping problem



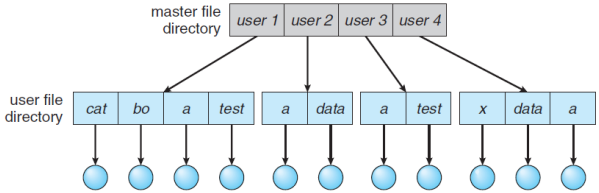
Two-Level Directory

- Separate directory for each user.



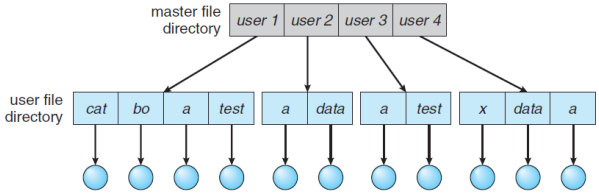
Two-Level Directory

- Separate directory for each user.
- Can have the same file name for different users.



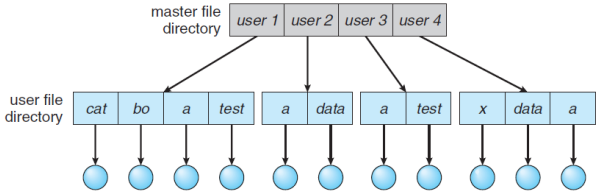
Two-Level Directory

- Separate directory for each user.
- Can have the same file name for different users.
- Efficient searching



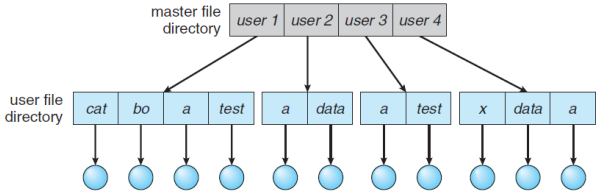
Two-Level Directory

- Separate directory for each user.
- Can have the same file name for different users.
- Efficient searching
- Path name: two level path, e.g., `/userB/file.txt`



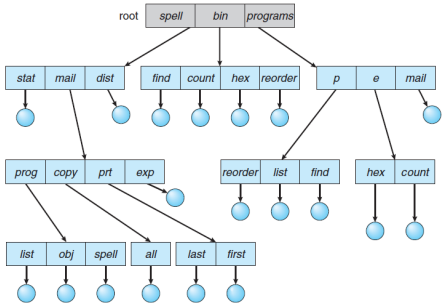
Two-Level Directory

- Separate directory for each user.
- Can have the same file name for different users.
- Efficient searching
- Path name: two level path, e.g., `/userB/file.txt`
- No grouping capability



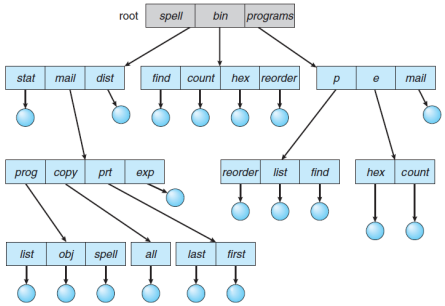
Tree-Structured Directories

- Efficient searching and grouping capability



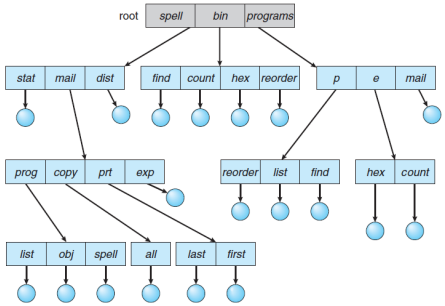
Tree-Structured Directories

- Efficient searching and grouping capability
- Current directory (working directory)
  - `cd /spell/mail/prog`



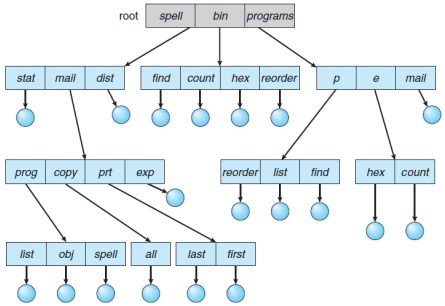
Tree-Structured Directories

- Efficient searching and grouping capability
- Current directory (working directory)
  - `cd /spell/mail/prog`
- Two types of path names:



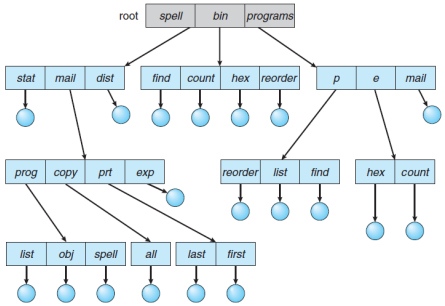
Tree-Structured Directories

- Efficient searching and grouping capability
- Current directory (working directory)
  - `cd /spell/mail/prog`
- Two types of path names:
  - Absolute path name: a path from the root.



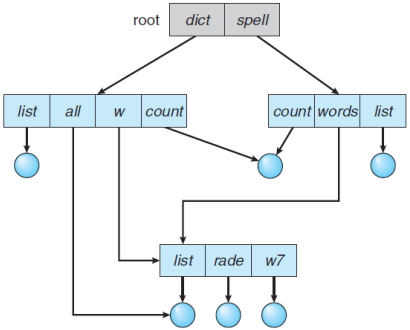
Tree-Structured Directories

- Efficient searching and grouping capability
- Current directory (working directory)
  - `cd /spell/mail/prog`
- Two types of path names:
  - Absolute path name: a path from the root.
  - Relative path name: a path from the current directory.



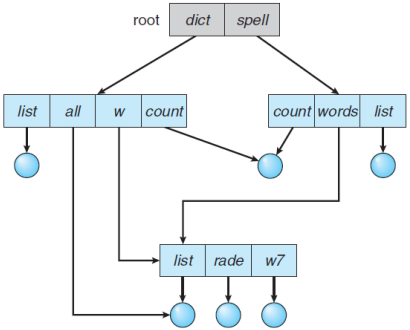
Acyclic-Graph Directories (1/3)

- Have shared subdirectories and files.



Acyclic-Graph Directories (1/3)

- Have shared subdirectories and files.
- Only one actual file exists with a shared file., so any changes made by one person are immediately visible to the other.



Acyclic-Graph Directories (2/3)

- Two approaches to implement shared files.

Acyclic-Graph Directories (2/3)

- Two approaches to implement shared files.
- 1. Duplicate all information about the file.

Acyclic-Graph Directories (2/3)

- Two approaches to implement shared files.
- 1. Duplicate all information about the file.
  - Both entries are identical and equal.
  - Consistency?

Acyclic-Graph Directories (2/3)

- Two approaches to implement shared files.
- 1. Duplicate all information about the file.
  - Both entries are identical and equal.
  - Consistency?
- 2. Link: another name (pointer) to an existing file.

Acyclic-Graph Directories (2/3)

- Two approaches to implement shared files.
- 1. Duplicate all information about the file.
  - Both entries are identical and equal.
  - Consistency?
- 2. Link: another name (pointer) to an existing file.
  - Resolve the link: follow pointer to locate the file.

Acyclic-Graph Directories (3/3)

- Deletion possibilities?

Acyclic-Graph Directories (3/3)

- Deletion possibilities?
- Remove the file content whenever anyone deletes it.

Acyclic-Graph Directories (3/3)

- Deletion possibilities?

- Remove the file content whenever anyone deletes it.
  - Dangling pointers: pointing to the nonexistent file.

Acyclic-Graph Directories (3/3)

- Deletion possibilities?

- Remove the file content whenever anyone deletes it.
  - Dangling pointers: pointing to the nonexistent file.
  - What if the remaining file pointers contain actual disk addresses?

Acyclic-Graph Directories (3/3)

- Deletion possibilities?

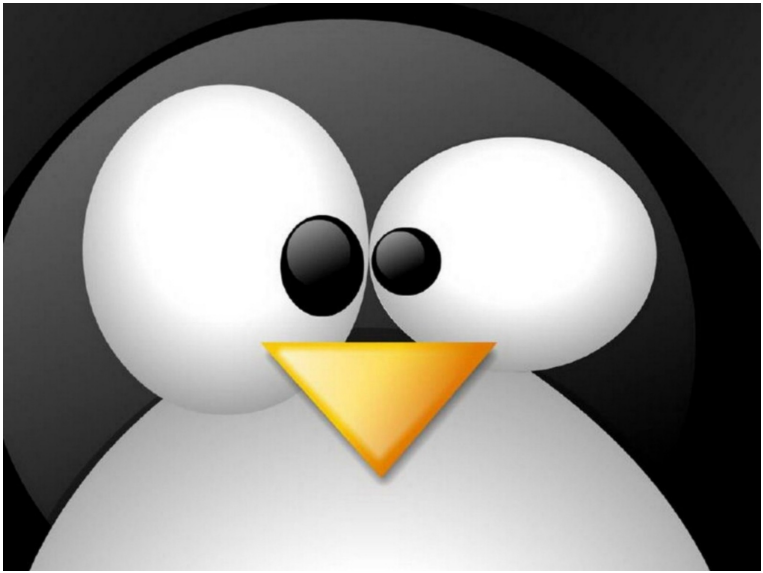
- Remove the file content whenever anyone deletes it.
  - Dangling pointers: pointing to the nonexistent file.
  - What if the remaining file pointers contain actual disk addresses?
  - Easy with soft-links (symbolic links)

Acyclic-Graph Directories (3/3)

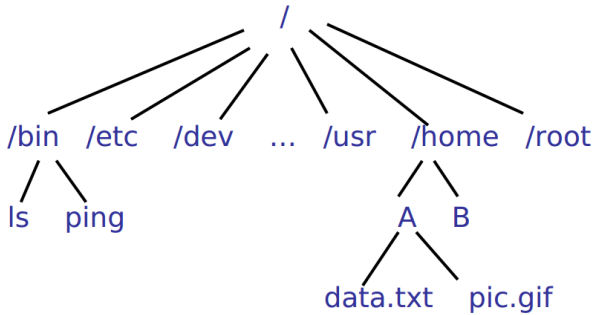
- Deletion possibilities?

- Remove the file content whenever anyone deletes it.
  - Dangling pointers: pointing to the nonexistent file.
  - What if the remaining file pointers contain actual disk addresses?
  - Easy with soft-links (symbolic links)

- Preserve the file until all references to it are deleted.
  - Hard links



Linux File System



/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 (`gzip`, `gunzip`)

/sbin

- Hold essential maintenance or system programs:
  - `fsck, fdisk, mkfs, shutdown, init, ...`

/sbin

- Hold essential maintenance or system programs:
  - `fsck, fdisk, mkfs, shutdown, 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, ...`

/home and /root

- The /home directory: the home directories for all users.
- The /root directory: the home directories for root user.

/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: hold temporary files or files with constantly varying content.

/tmp and /var

- The `/tmp` and `/var` directories: hold temporary files or files with constantly varying content.
- The `/tmp` directory: files that only need to be used briefly and can afford to be deleted at any time.

/tmp and /var

- The `/tmp` and `/var` directories: hold temporary files or files with constantly varying content.
- The `/tmp` directory: files that only need to be used briefly and can afford to be deleted at any time.
- The `/var` directory: a bit more structured than `/tmp`.

/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

/proc

- It is a virtual file system
- Provided by the kernel
- Provides information about the kernel and processes.

File and Directory Management

- `getcwd()` returns the current working directory.
- `chdir()` changes the current working directory to `path`

```c
#include <unistd.h>

char *getcwd(char *buf, size_t size);
int chdir(const char *path);
```

File and Directory Management

- `mkdir()` creates the directory path.

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

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

File and Directory Management

- `mkdir()` creates the directory path.

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

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

- `rmdir()` removes a directory from the filesystem.

```c
#include <unistd.h>

int rmdir(const char *path);
```

File and Directory Management

- `opendir()` creates a directory stream representing.
- `readdir()` returns the next entry in the directory.
- `closedir()` closes the directory stream.

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

DIR *opendir(const char *name);;
struct dirent *readdir(DIR *dir);
int closedir(DIR *dir);
```

File System Commands (1/3)

- `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

File System Commands (2/3)

- `mv`: moves files
- `cp`: copies files
- `rm`: removes directory
- `gzip/gunzip`: to compress and uncompress a file
- `tar`: to compress and uncompress a file
- `e2fsck`: check a Linux ext2/ext3/ext4 file system

File System Commands (3/3)

- `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

File Sharing and Protection

File Sharing

- Sharing of files on multi-user systems is desirable.

File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a protection scheme.

File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a protection scheme.
  - User IDs identify user

File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a protection scheme.
  - User IDs identify user
  - Owner of a file/directory: the user who can change attributes and grant access and who has the most control over the file.

File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a protection scheme.
  - User IDs identify user
  - Owner of a file/directory: the user who can change attributes and grant access and who has the most control over the file.
  - Group of a file/directory: a subset of users who can share access to the file.

Access Lists and Groups (1/2)

- Mode of access: read, write, execute (`rwx`)

Access Lists and Groups (1/2)

- Mode of access: read, write, execute (`rwx`)
- Three classes of users:

Access Lists and Groups (1/2)

- Mode of access: read, write, execute (`rwx`)
- Three classes of users:
  - Owner: the user who created the file.

Access Lists and Groups (1/2)

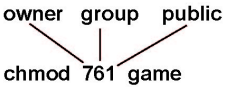
- Mode of access: read, write, execute (`rwx`)
- Three classes of users:
  - Owner: the user who created the file.
  - Group: a set of users who are sharing the file and need similar access.

Access Lists and Groups (1/2)

- Mode of access: read, write, execute (`rwx`)
- Three classes of users:
  - Owner: the user who created the file.
  - Group: a set of users who are sharing the file and need similar access.
  - Universe: all other users in the system.

Access Lists and Groups (1/2)

- Mode of access: read, write, execute (`rwx`)
- Three classes of users:
  - Owner: the user who created the file.
  - Group: a set of users who are sharing the file and need similar access.
  - Universe: all other users in the system.
- Owner access `rwx`: 111 (7)
  Group access `rwx`: 110 (6)
  Public access `rwx`: 001 (1)



Access Lists and Groups (2/2)

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 jwg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2012	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2012	program
drwx--x--x	4 tag	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Summary

Summary

- File concept: types, attributes, operations, locks

Summary

- File concept: types, attributes, operations, locks
- Access methods: sequential, direct

Summary

- File concept: types, attributes, operations, locks
- Access methods: sequential, direct
- Directory structure: single-level, two-level, tree-structured, acyclic-graph, general-graph

Summary

- File concept: types, attributes, operations, locks
- Access methods: sequential, direct
- Directory structure: single-level, two-level, tree-structured, acyclic-graph, general-graph
- File sharing and protection: rwx, owner, group, universe

Questions?

Acknowledgements

Some slides were derived from Avi Silberschatz slides.