Introduction to UNIX Basic Concepts

EE1356 Introduction to Information Systems

Why Learn UNIX?

- Powerful and flexible
  - > 30 years of development
- Programmer’s environment
  - A toolset for hackers
  - Open standards
- Runs research/design tools
  - Run the internet (various servers)
  - IC design tools
  - Abundant open source/free tools

Philosophy of UNIX

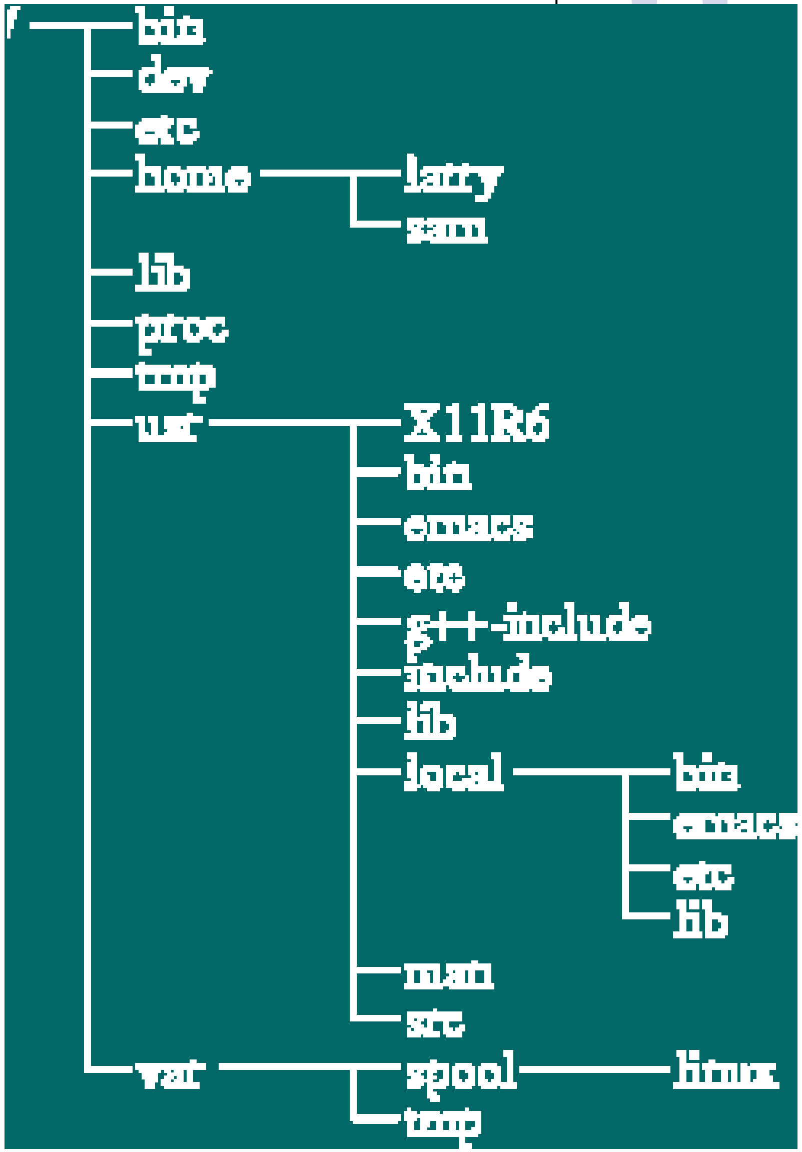
- A collection of interoperable tool sets
  - Small tools work together
  - We can build specific tools without re-inventing wheels
- Work in command line mode
  - You have to type commands in text console
  - Have longer learning curve
  - Initially it seems difficult for new users to remember commands
  - Eventually it will pay off: faster and flexible

All Things are Files

- In UNIX, files are
  - Collection of data
  - Program interface (standard IO)
  - Device drivers
  - Kernel interface (/proc)
  - ETC
- A powerful concept
  - Program capable of handling files can be extend to handle information beyond original designs as long as we can model the information as files.

File Systems

- File systems are structured storage for organizing files.
- Users usually view a file system as hierarchical structure.
  - A Tree of directories.
- A directory in the tree is identified by paths: directory names concatenated with “/”.
  - For example, /usr/local/bin is the directory “bin” under “/usr/local”, and “/usr/local” is the “local” directory under “/usr”.
  - Actually “/usr” is a directory under “/”, which we call “root” directory.



More On Path Names

- Paths begins with “/” (absolute path name)
  - Starting hierarchy from the root dir.
- Path begins with a “~”
  - “~/” means your home dir.
  - “~username” mean is home fir of username.
- Other path types (relative path name)
  - If you are at /usr, then local/lib mean /usr/local/lib
  - Path starting with “./”, it is relative to the current dir. We usually “./” to specify the execution file in the current dir. For example, “./some_binary”.
  - Path starting with “../”, it is relative to the parent dir.
  - Paths can be constructed with multiple “./” and “../”. For example, “../../” is the grand-parent of current dir.

About File and Directory Names

- In general, a filename can contain any character except “/” (and “0” ASCII code), and is limited to 256 characters.
  - In DOS, we use “/”.
- Unlike DOS, not directly related to the physical disk/partition layout
  - Actually, directories may not even reside in the current computer as they are “mounted” from other networked computers.
- Note the case sensitivity – Mozilla and mozilla are not the same!
- File names don’t have to include extension such as “.exe” in DOS for executables, but it is usually good idea to use file extension according to conventions.
  - For example, “.c” for C files, “.jpg” for JPEG image files, “.txt” for text file, etc.

A Multi-user System

- UNIX is a true **multi-user** system
- Every users in the system share the same hardware but think they own the system solely.
  - You will notice the system’s response slow down if two many users login at the same time.
- Implement a system with stronger security
  - No user is allowed to intrude others’ work
  - Failed user programs will not interfere system function

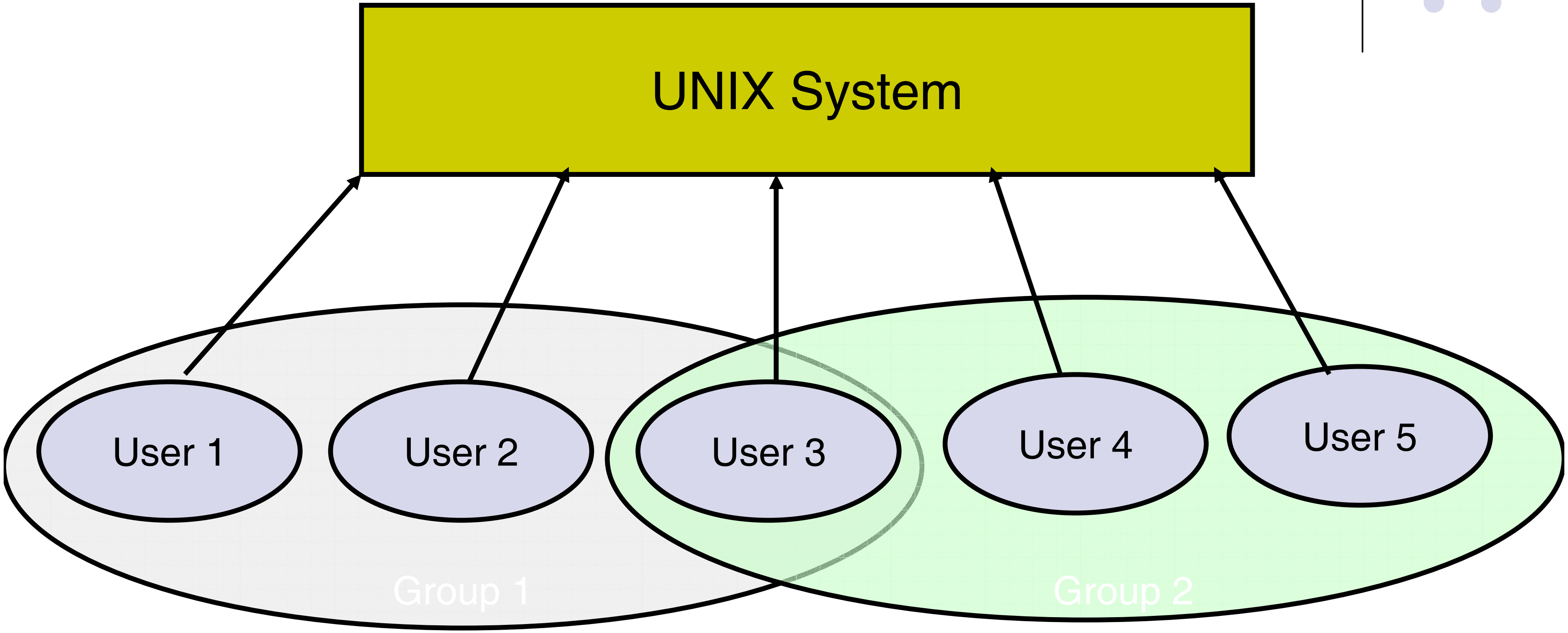
Files Have Owners

- In multi-user system, files are therefore maintained by their owners.
- Users have a home directory for storing their own data.
- “root” user is responsible for system files.

User Account

- To use your UNIX machine, you must have an **account** on the machine.
- Account login
  - login name (login:)
  - password (password:)
- After login, you will be located at your home directory.
  - For example, /home/loginname.

Users and Groups



- A group of users can have the same permission on assigned on certain files.
- Users can belong to multiple groups
- Users and groups are identified by numbers: User ID (UID) and Group ID (GID)

File ownership and Permissions

- Every file is the exclusive property of one user and one group.
- File permissions can be set for
  1. owner (i.e., user itself)
  2. owner group
  3. other
- Three permission properties
  1. Read (r)
  2. Write (w)
  3. eXecute (x)
- There are 3 permission properties for 3 types of users (owner, group, other).

Permission Properties

- Read (r):
  - Enables a user to read the contents of a file.
  - For a directory, the user can list its contents (i.e. the files in this directory).
- Write (w):
  - Allows the modification of a file's contents.
  - For a directory, allows a user to add or remove files from this directory, even if he/she is not the owner of these files.
- eXecute(x):
  - Enables a file to be executed (only executable files normally have this permission set).
  - For a directory, it allows a user to **traverse** it, which means going into or through that directory.
  - Note that this is different from the read access: you may be able to traverse a directory but still be unable to read its contents!

Your First Command: ls

- “ls –l”
  - “ls” is the command to **lis**t files and directories.
  - “-l”
    - one of the option given to “ls”
    - print all information about files and directories.
  - Option switch
    - To set the mode for a command. For example, the display format of “ls”.
    - Usually specified by a “-”.

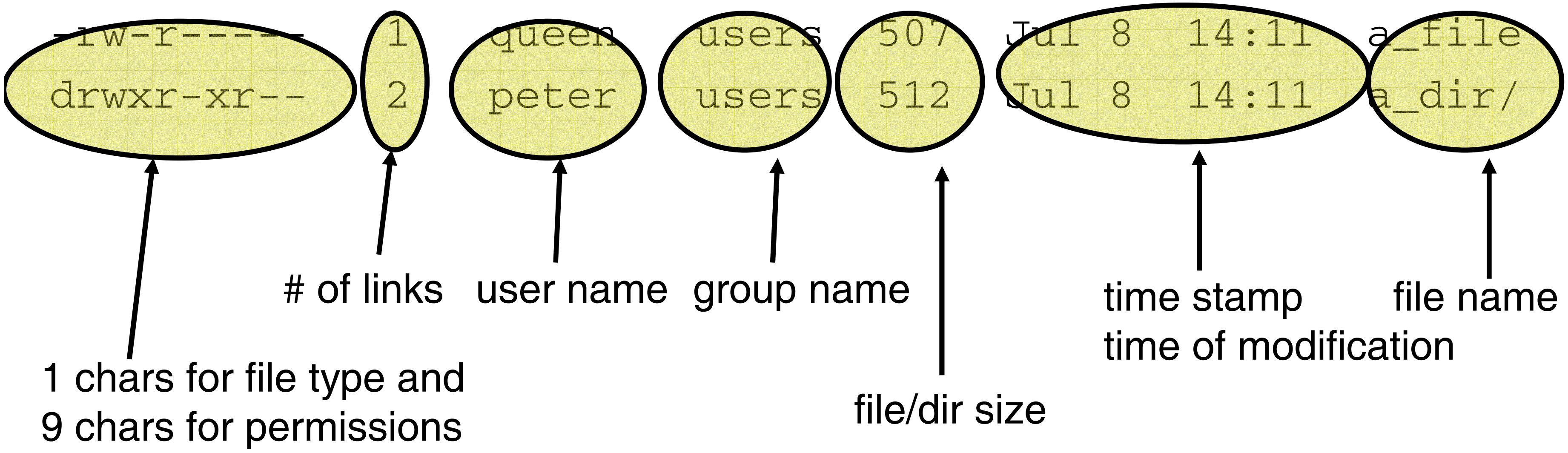
Sample “ls –l” outputs

```
$ ls -l
total 1
-rw-r----- 1 queen users 0 Jul 8 14:11 a_file
drwxr-xr-- 2 peter users 1024 Jul 8 14:11 a_dir/
$
```

- “$” is the shell prompt. You can customize this by configuring your shell.
- “ls –l” is command you typed followed by and “Enter” key.
- “total 1” means total block of files listed is one.
  - A block is 512Byte.

Sample “ls –l” outputs (Cont.)

```
$ ls -l
total 1
-rw-r-----   1   queen   users   507   Jul 8   14:11   a_file
drwxr-xr--   2   peter   users   512   Jul 8   14:11   a_dir/
```



# of links

user name

group name

time stamp

file name

time of modification

1 chars for file type and
9 chars for permissions

file/dir size

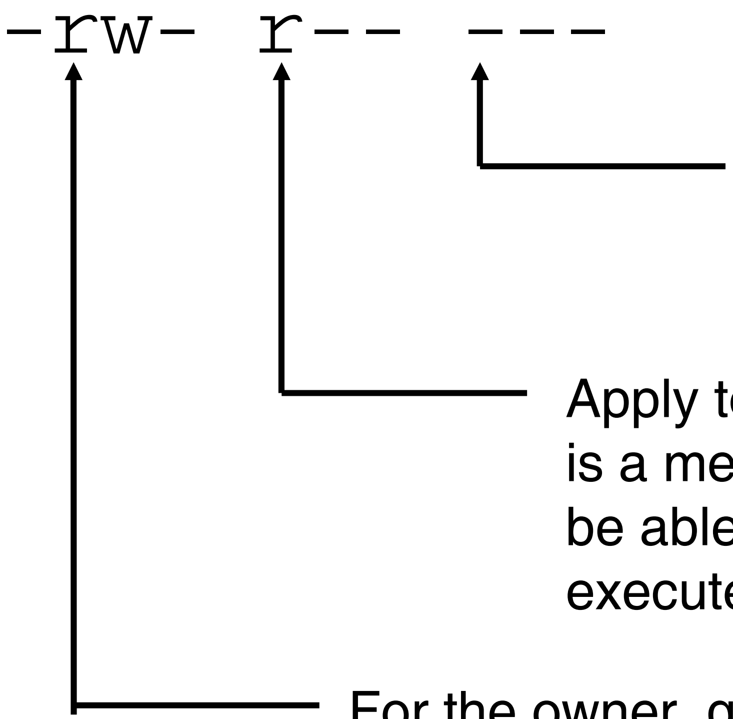
FYI, all these information are stored on disk with a data structure called “inode”.

Explaining “ls –l” outputs for files

```
-rw-r----- 1 queen users 0 Jul 8 14:11 a_file
```

- File type: **-** for an ordinary file, **d** for a directory
- Permission bits: A dash (-) means that the permission is not set. As indicated earlier, we can set file permission for owner (yourself), users in a group and others. And for each, there are permissions for read, write and execution (a total of 9 bits).

File Permission Exmples



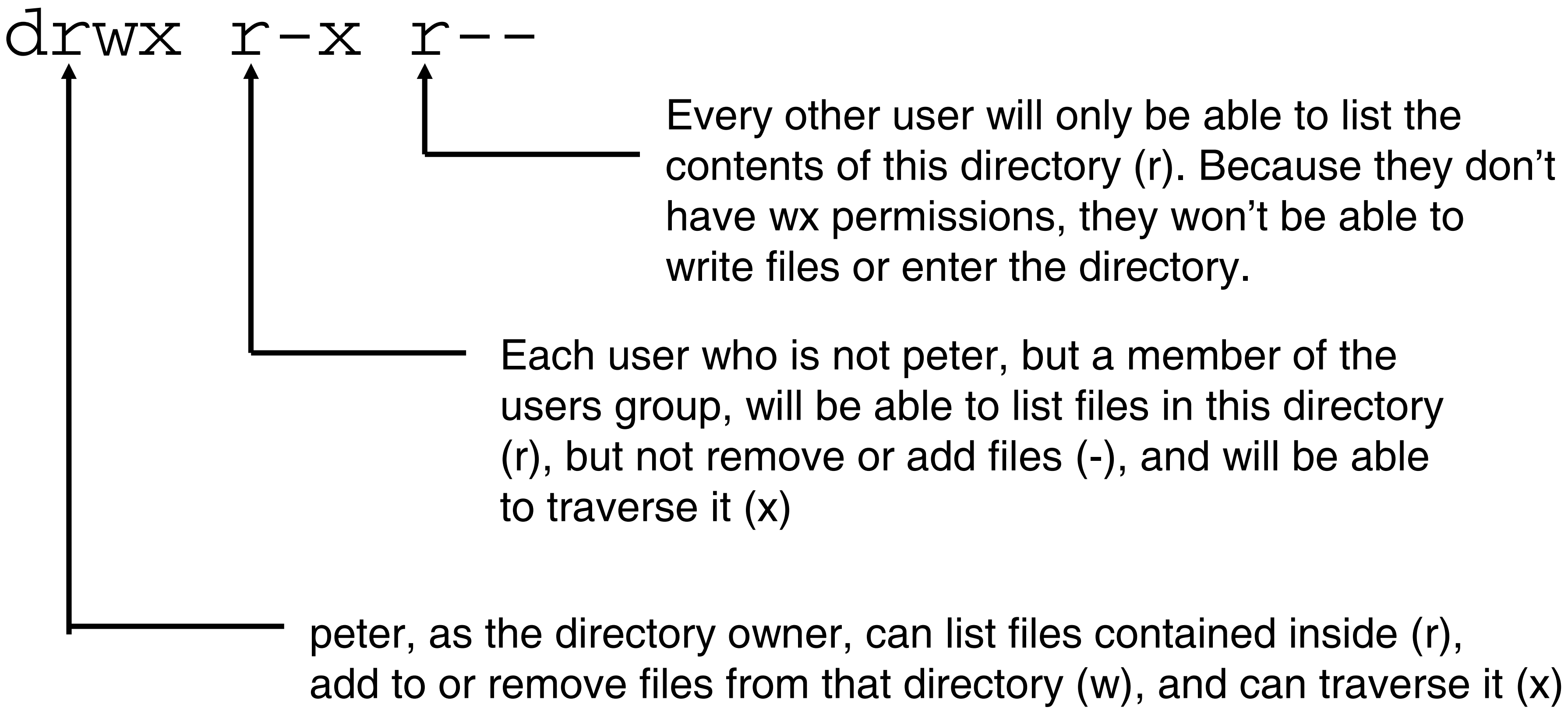
```
-rw- r-- ---
```

Apply to any user who is not queen and is not a member of the users group. Those users won’t have any rights on the file at all.

Apply to any user who is not queen but who is a member of the users group. They will be able to read the file (r), but not write or execute it (--)).

For the owner, queen, has the right to read the file (r), to modify its contents (w) but not to execute it (-).

Directory Permission Exmples



```
drwx r-x r--
```

Every other user will only be able to list the contents of this directory (r). Because they don’t have wx permissions, they won’t be able to write files or enter the directory.

Each user who is not peter, but a member of the users group, will be able to list files in this directory (r), but not remove or add files (-), and will be able to traverse it (x)

peter, as the directory owner, can list files contained inside (r), add to or remove files from that directory (w), and can traverse it (x)

Note: “root” user has the capability to change every permission of any files in the system.

You Second Command: man

- To know more about “ls”, type “man ls”
- “man” stands for **man**ual.
- Learn to read reference manual is essential in learning UNIX commands.
- Each reference manual contains several parts of information about the commands
  - NAME, SYNOPSIS, DESCRIPTION, OPTIONS, ENVIRONMENT VARIABLES, FILES, SEE ALSO
  - Some also have examples and usage notes
- “man –k keyword” can list commands with keywords in the reference manual.
  - Only works if “root” has installed the index files.