# Server Operating Systems

# Lecture 4
# Advanced File and Directory Mangement

# Copying Files

The **cp** command is used to copy a file or files to a new location.

Can copy file(s) to a new location in the same directory.

```
cp source_file destination_file
```

Cannot have two files named the same in the same directory.

# Copying Files

Can copy file(s) to another directory anywhere in the file system (with correct permissions).

```
cp source_file(s) destination_directory
```

Copies the source files to the destination directory.  The directory is created if it doesn’t exist.

Source files keep their original name unless explicitly renamed.

Metacharacters can be used in file names.

```
cp  image*  /home/user01
```

# How Not to “Clobber” a File

In Unix, “clobber” means to overwrite a file.

– This is what will happen normally, *without warning*, if you copy a file to a destination where a file of that name already exists.

The “interactive” option prevents clobbering.

```
cp –i source_file destination_file

cp –i source_file(s) destination_directory
```

– User is prompted for confirmation *only* when a file is about to be clobbered.

# Copying Directories

Use the “recursive” option to copy directories.

```
cp –r[i] source_directory(s) destination_directory
```

– Directories that do not exist will be automatically created.

As before, you can copy directories to a new location in the same directory or in another location in the file system.

# Renaming Files

B

Files can be renamed and moved with the same command, **mv.**

Renaming a file in the current directory.

```
mv [-i] source_file target_file
```

– Changes the name of the source file, no copies are made.

# Moving Files

Moving a file to another directory.

```
mv [-i] source_file(s) target_dir
```

- Moves the source file(s) to the target directory.  The files only exist in the target directory after issuing the command.
- When moving a single file, giving a new file name at the end of the target directory will move *and* rename the file.

# Renaming and Moving Directories

Renaming a directory within the current directory.

```
mv [-i] source_directory target_name
```

Moving a directory and its contents.

```
mv [-i] source_directory target_directory
```

- If the target directory exists, the source directory is moved into it.
- If the target directory does not exist, the source directory is just renamed.

# Input/Output

c

To tell the UNIX operating system what to do, it has to get input from the user.

Everything that is output by the UNIX operating system has to “go” somewhere for a user to know what it did.

Input is normally gotten from the keyboard while output usually goes to the monitor.

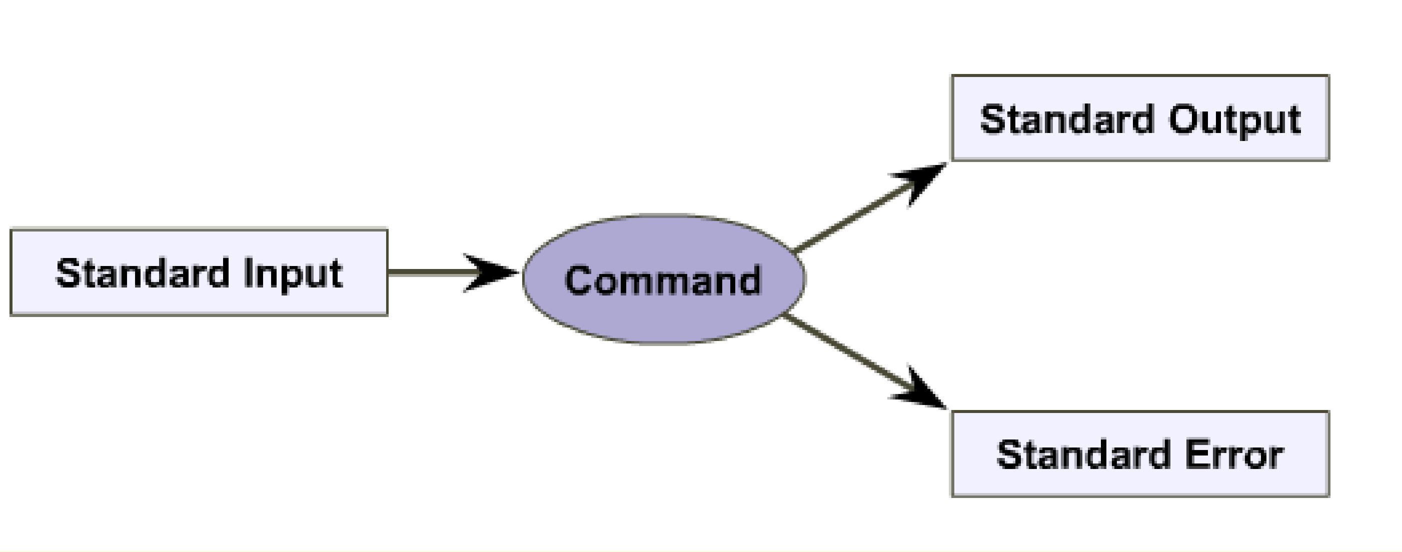
The Unix OS calls these locations:

*stdin*, *stdout*, and *stderr*.

# Input/Output

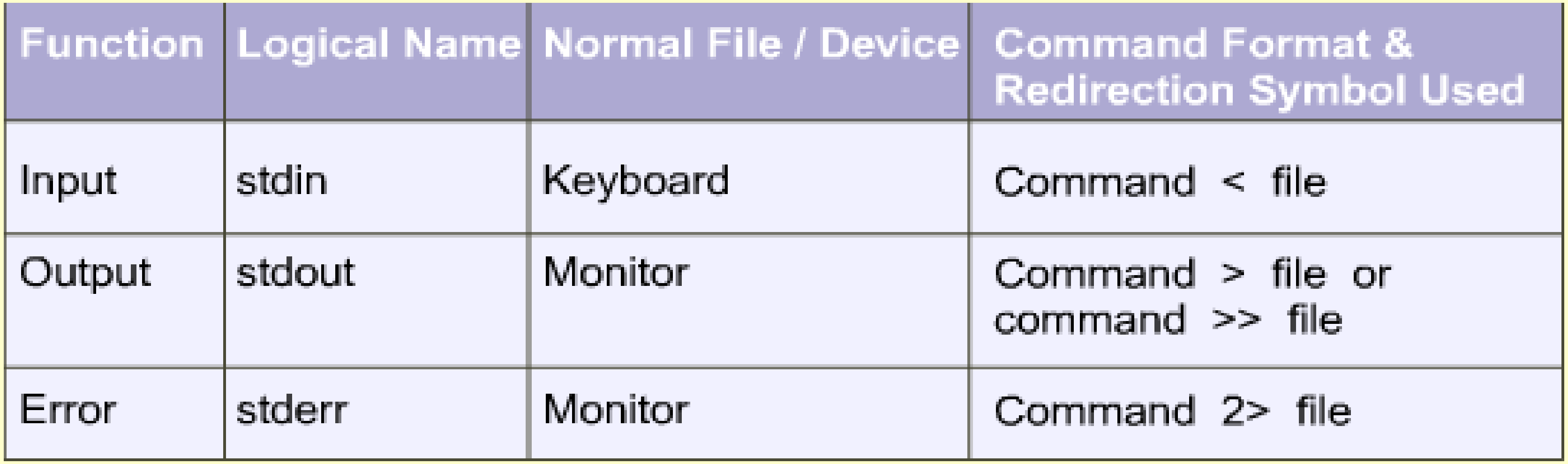
A UNIX command is a process that the operating system carries out.

It also uses the standard I/O channels on the system.



# Input/Output Redirection

Unix provides a way to *redirect* the input to, and output from, a command.



| Function | Logical Name | Normal File / Device | Command Format & Redirection Symbol Used |
|---|---|---|---|
| Input | stdin | Keyboard | Command < file |
| Output | stdout | Monitor | Command > file or command >> file |
| Error | stderr | Monitor | Command 2> file |

# Redirection

```
ls -l > listoutput.txt
```

will redirect the output from a list command into a text file.

```
rm report*.doc  2>  errorlist.txt
```

will redirect any error messages from the remove command into a text file

```
mail < report.doc
```

will cause the mail command to read in the file report.doc as input

```
ls -l >> textfile.txt
```

will append the output from the ls command to the end of the contents of textfile.txt.

# Input/Output Redirection Tips

Redirecting output to a file creates the file if it does not exist or clobbers (overwrites) it if it does.

- Use `set –o noclobber` to enable “no clobbering”.
- Use `set +o noclobber` to disable “no clobbering”.
- Use `>|` to override “no clobbering” on a one-off basis.

Can use >> with `cat` to combine multiple files into one.

```
cat part1 part2 part3 >> book.doc
```

# Command Piping ( | )

Command piping passes the standard output of one command to the standard input of another.

redirect takes the output of a command to a file;

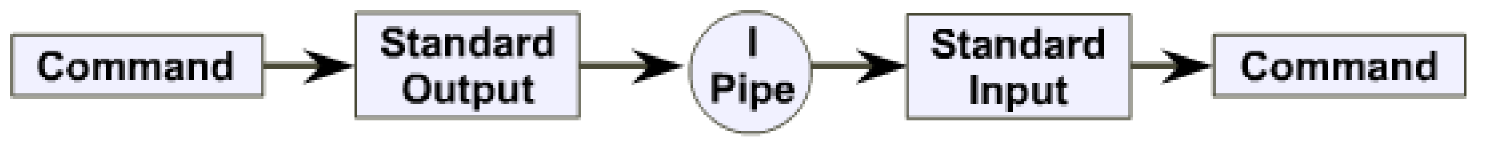
pipe takes the ouput of a command to another command

```
command1 | command2
```

A very common use for the pipe is: `ls -l | more.`

For an administrator to see the users on the system, he or she would type:

```
cat /etc/passwd | more
```



# mv Exercise

```
$ mv  ??[abc]  ../..
```

# mv Exercise

```
$ mv  ??[abc]  ../..
```

Move any file(s) with only three characters in the name, up two directory levels

The filenames can have anything in the first two character positions, with a or b or c in the third position,