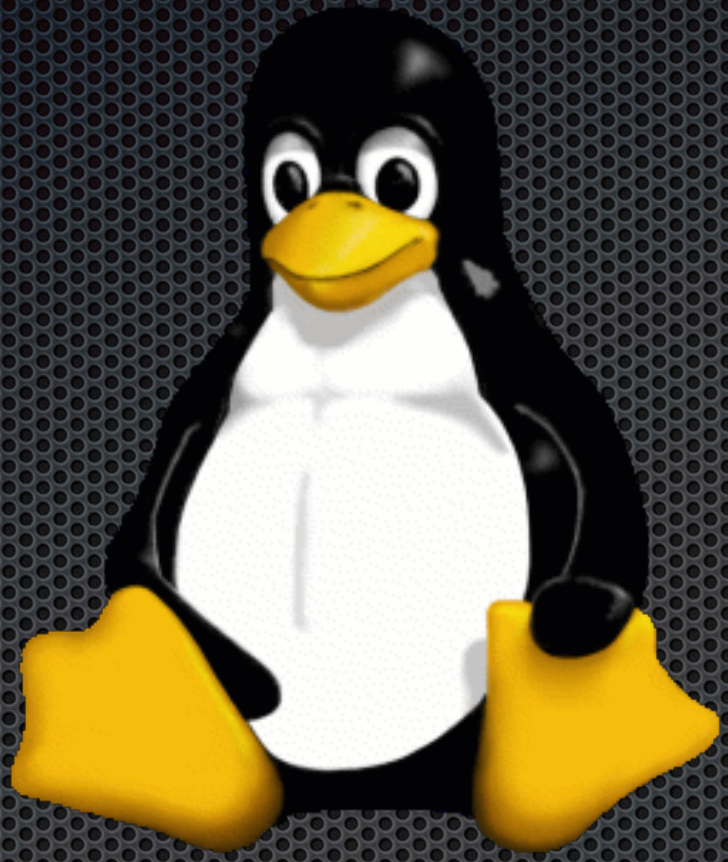
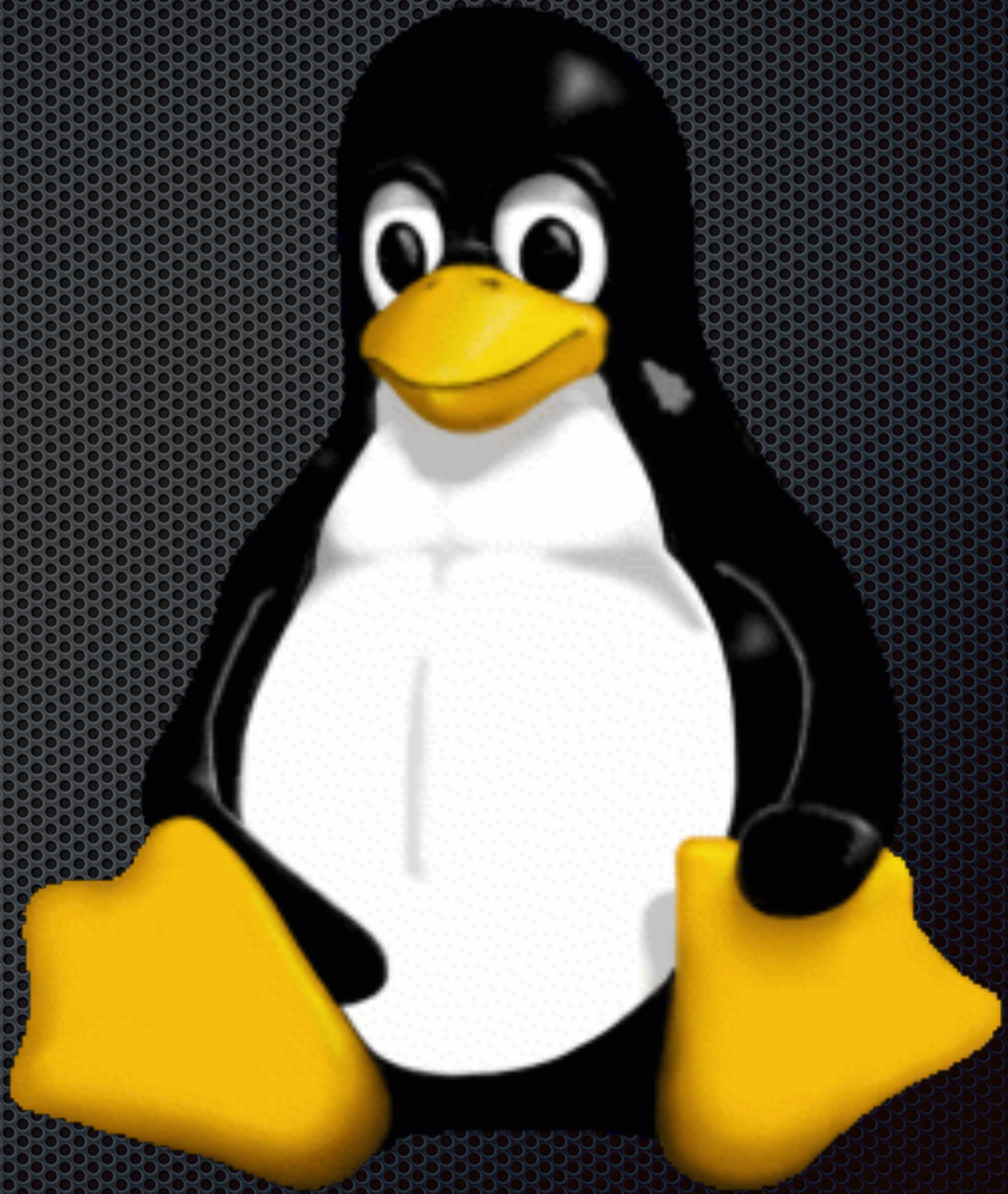


Introduction to Linux

What is Linux?



Linux

- Free operating system
- Open source project
- Began in 1991
- Run on most server worldwide
- Many different versions
    - Called flavors, distributions, or distros

"I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)…"

–LINUS TORVALDS ON COMP.OS.MINIX IN 1991

What We're Doing Today

Log into a remote machine with ssh

Navigate the Linux shell

Create/Move/Copy files

Run/Manage processes

Piping/Redirection

Log in to LNX01

Open the terminal

(if Windows - Use Putty)

Type the following:

```
ssh <uniqueid>@ceclnx01.cec.miamioh.edu
```

Press enter

Enter your Miami password

You are now logged into the server at ceclnx01.cec.miamioh.edu

```
user@ceclnx01:~/$ echo “Hello, World!”
```

Anatomy of a Linux Command

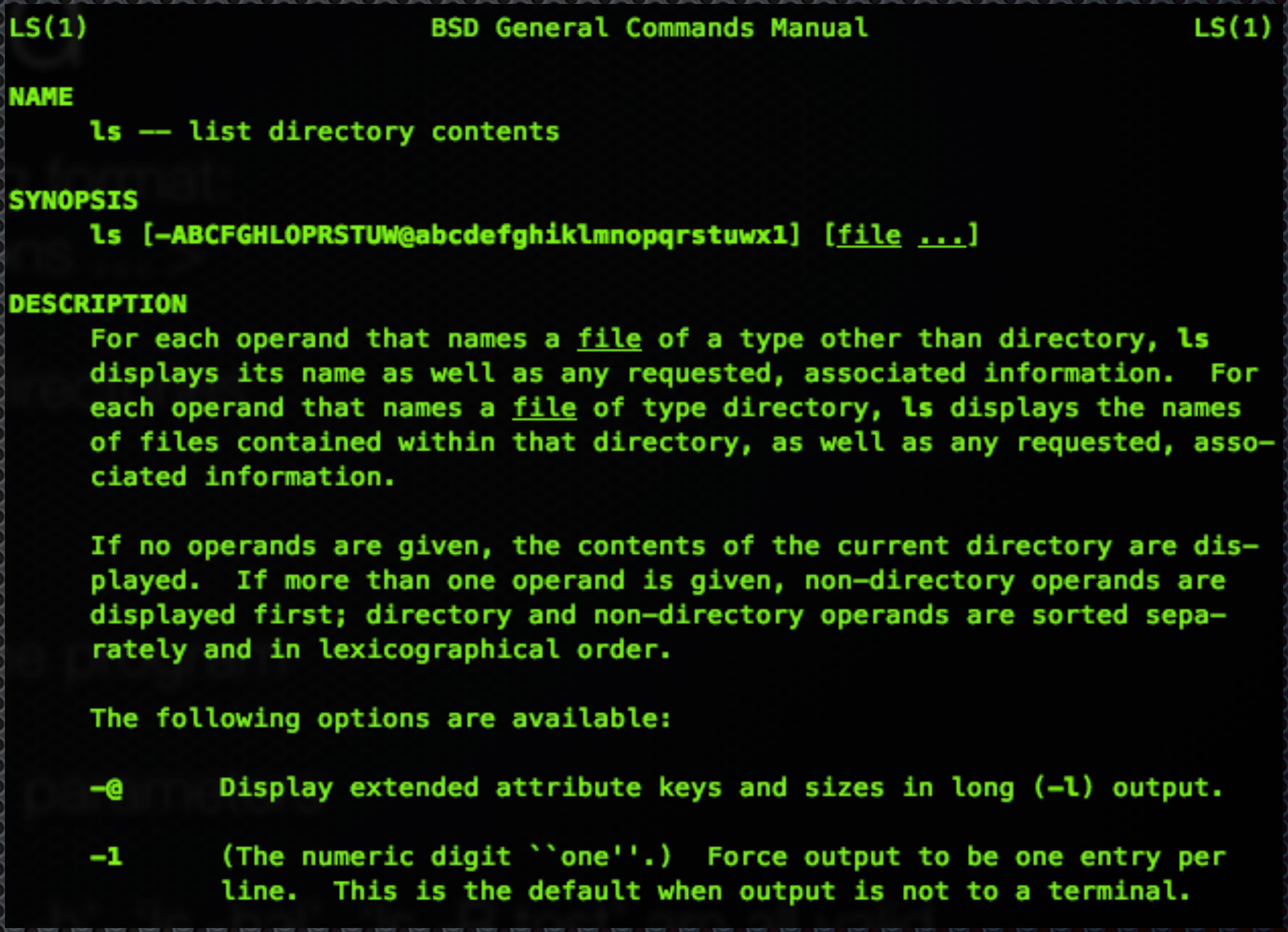
Follows the following format:
<command> <options … >

Example - list files/directories

ls [-halR] [file …]

'ls' is the name of the program

[ ] indicates optional parameters



```
LS(1)                     BSD General Commands Manual                    LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls
     displays its name as well as any requested, associated information.  For
     each operand that names a file of type directory, ls displays the names
     of files contained within that directory, as well as any requested, asso-
     ciated information.

     If no operands are given, the contents of the current directory are dis-
     played.  If more than one operand is given, non-directory operands are
     displayed first; directory and non-directory operands are sorted sepa-
     rately and in lexicographical order.

     The following options are available:

     -@      Display extended attribute keys and sizes in long (-l) output.

     -1      (The numeric digit ``one''.)  Force output to be one entry per
             line.  This is the default when output is not to a terminal.
```

For example: 'ls', 'ls -h', 'ls -hal', 'ls -R test' are all valid commands

Many other parameters - **Use man <command> to read the manual**

Press 'q' to exit the manual

First - Populate Your Directory

Run the following commands ***exactly*** as shown

```
wget http://www.users.miamioh.edu/rogerskw/acm.tar
```

```
tar -xvf acm.tar
```

This downloads an archive from my website to your directory. Then it extracts the archive

Special Directory Names

/	root directory
~/	User's home directory
./	Current directory
../	Up 1 directory
~username/	username's home

Move Around

Command	Action
ls [-Rahl] [file …]	List files in current or specified directory
cd [directory]	Navigate to a directory. If not provided, assumes ~/ .. will go one directory up
mkdir [newdirectory]	Create a new directory with name 'newdirectory'

What We're Doing Today

~~Log into a remote machine with ssh~~

~~Navigate the Linux shell~~

Create/Move/Copy files

Run/Manage processes

Piping/Redirection

Create/Move/Delete/Copy Files and Directories

Command	Action
touch <filename>	Create a new (empty) file with name filename
mkdir <dirname>	Create a directory with name dirname
rm [-r] <name>	Remove file named name. -r to remove recursively
mv <filename> <newloc>	Move filename to newloc.
cp <filename> <newloc>	Copies filename to newloc

Wildcards

Use an asterisk (*) for a wildcard

Ex: run the following commands:

```
cp inputsdir/*.txt.bak . # <— The last . is important! (why?)
ls *.txt.bak
rm *.txt.bak
```

What does a wildcard do?

Challenge!

Move all text files to a new directory called
<youruniqueid>dirs

Copy all text files to this directory

Go to this directory and remove all .txt files that begin with an `E` and end with a `t` (not the file extension)

What We're Doing Today

~~Log into a remote machine with ssh~~

~~Navigate the Linux shell~~

~~Create/Move/Copy files~~

Run/Manage processes

Piping/Redirection

Run a Program

Compile a C program

```
gcc HelloWorld.c -o HelloWorld
```

Run your program

```
./HelloWorld
```

Why is the ‘ ./ ‘ necessary?

Most commands that you use are actually programs

Running in the Background

Some commands take a long time to complete (or even never complete)

run **./sleepforever**

- You can't do other commands until it completes
- Press **Control-C** to end the process

Run it in *the background* with **./sleepforever &**

- Try to run another command (ex: ls)

View Processes in the Background

View the manual for **ps** (man ps)

- Reminder: press ‘q’ to exit the manual

Use ps to see the processes currently running

- You should see **sleepforever** listed

Get a real time view with **top**

Killing Processes

The first column is the PID - Process ID

This is the unique identifier for the process

View the manual for **kill**

Kill a running process with **kill <processid>**

Kill sleepforever now

Check out **pkill** for killing processes by their names

Switch Between Background and Foreground

1. Run sleepforever again - not in the background
    1. Remember, CTRL-C *killed* the process
2. Use **CTRL-Z** to *stop* the process
    1. The program is still alive, but is *not* running
3. Bring it back to the *foreground* with **fg**
4. Stop the process again
5. Run the process in the *background* with **bg**
6. Kill the process with ps and kill

What We're Doing Today

~~Log into a remote machine with ssh~~

~~Navigate the Linux shell~~

~~Create/Move/Copy files~~

~~Run/Manage processes~~

Piping/Redirection

Redirection

- You can write the output of a program to a file with **>**
  - Ex: **echo “y” > yes.txt**
  - View the contents with **cat yes.tx**t
  - You can **append** to the file with **>>**
- Try running **./mcdonalds**
- If a program takes input, you can use a file with preset input with **<**
  - Ex: **./mcdonalds < yes.txt**
- Common Use: Redirect to /dev/null silences the output of a process

Piping

Many times you will want to use the output of one process as the input to another process.

This is called **piping**. Do this with the **|** character

Ex: **echo “y” | ./mcdonalds**

You can chain many pipes together and even end them with redirecting to a file

Piping is an incredibly powerful tool that you will use frequently on Linux systems

LaTeX

with Dr. Rao

Next Workshop - 9/24