Minnesota Supercomputing Institute

UNIVERSITY OF MINNESOTA

Introduction to Linux

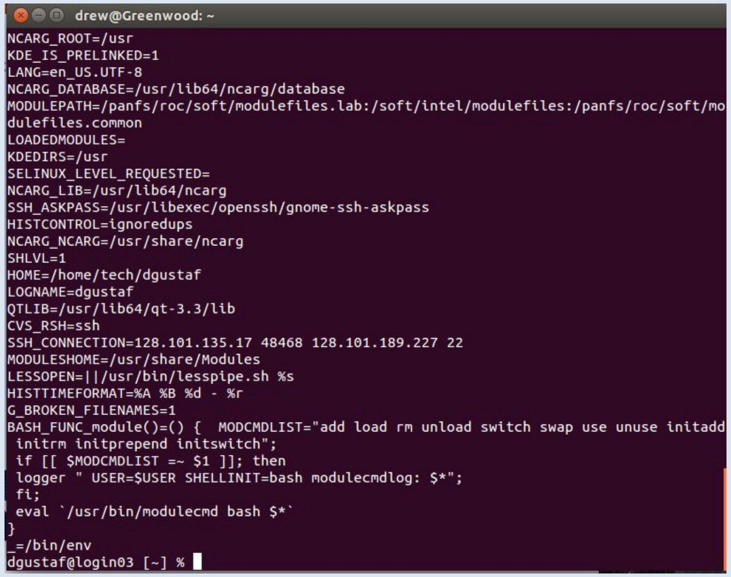
Andrew Gustafson

UNIVERSITY OF MINNESOTA

Minnesota Supercomputing Institute

What is Linux?

Linux is an open source Unix-like operating system.



drew@Greenwood: ~

```
NCARG_ROOT=/usr
KDE_IS_PRELINKED=1
LANG=en_US.UTF-8
NCARG_DATABASE=/usr/lib64/ncarg/database
MODULEPATH=/panfs/roc/soft/modulefiles.lab:/soft/intel/modulefiles:/panfs/roc/soft/mo
dulefiles.common
LOADEDMODULES=
KDEDIRS=/usr
SELINUX_LEVEL_REQUESTED=
NCARG_LIB=/usr/lib64/ncarg
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
NCARG_NCARG=/usr/share/ncarg
SHLVL=1
HOME=/home/tech/dgustaf
LOGNAME=dgustaf
QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
SSH_CONNECTION=128.101.135.17 48468 128.101.189.227 22
MODULESHOME=/usr/share/Modules
LESSOPEN=||/usr/bin/lesspipe.sh %s
HISTTIMEFORMAT=%A %B %d - %r
G_BROKEN_FILENAMES=1
BASH_FUNC_module()=() {  MODCMDLIST="add load rm unload switch swap use unuse initadd
 initrm initprepend initswitch";
 if [[ $MODCMDLIST =~ $1 ]]; then
 logger " USER=$USER SHELLINIT=bash modulecmdlog: $*";
 fi;
 eval `/usr/bin/modulecmd bash $*`
}
_=/bin/env
dgustaf@login03 [~] % 
```

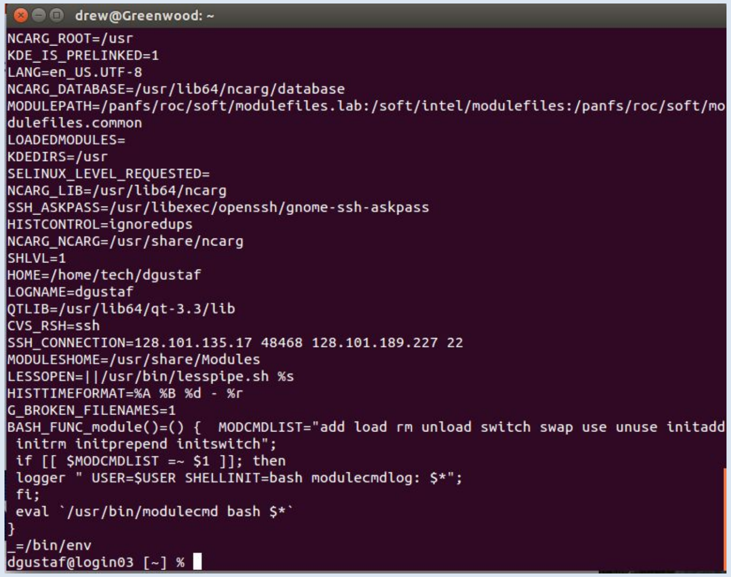
- Unix was developed at AT&T Bell Labs in the 1970's.
- Linux was developed by Linus Torvalds in the 1990's, and is now developed by a community as a free operating system.
- Linux is: free, stable, extensible, and fast.
- Unix/Linux are the dominant OSes for large compute systems.

UNIVERSITY OF MINNESOTA

Minnesota Supercomputing Institute

What is Linux?

Linux is an open source Unix-like operating system.



drew@Greenwood: ~

```
NCARG_ROOT=/usr
KDE_IS_PRELINKED=1
LANG=en_US.UTF-8
NCARG_DATABASE=/usr/lib64/ncarg/database
MODULEPATH=/panfs/roc/soft/modulefiles.lab:/soft/intel/modulefiles:/panfs/roc/soft/mo
dulefiles.common
LOADEDMODULES=
KDEDIRS=/usr
SELINUX_LEVEL_REQUESTED=
NCARG_LIB=/usr/lib64/ncarg
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
NCARG_NCARG=/usr/share/ncarg
SHLVL=1
HOME=/home/tech/dgustaf
LOGNAME=dgustaf
QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
SSH_CONNECTION=128.101.135.17 48468 128.101.189.227 22
MODULESHOME=/usr/share/Modules
LESSOPEN=||/usr/bin/lesspipe.sh %s
HISTTIMEFORMAT=%A %B %d - %r
G_BROKEN_FILENAMES=1
BASH_FUNC_module()=() {  MODCMDLIST="add load rm unload switch swap use unuse initadd
 initrm initprepend initswitch";
 if [[ $MODCMDLIST =~ $1 ]]; then
 logger " USER=$USER SHELLINIT=bash modulecmdlog: $*";
 fi;
 eval `/usr/bin/modulecmd bash $*`
}
_=/bin/env
dgustaf@login03 [~] % 
```

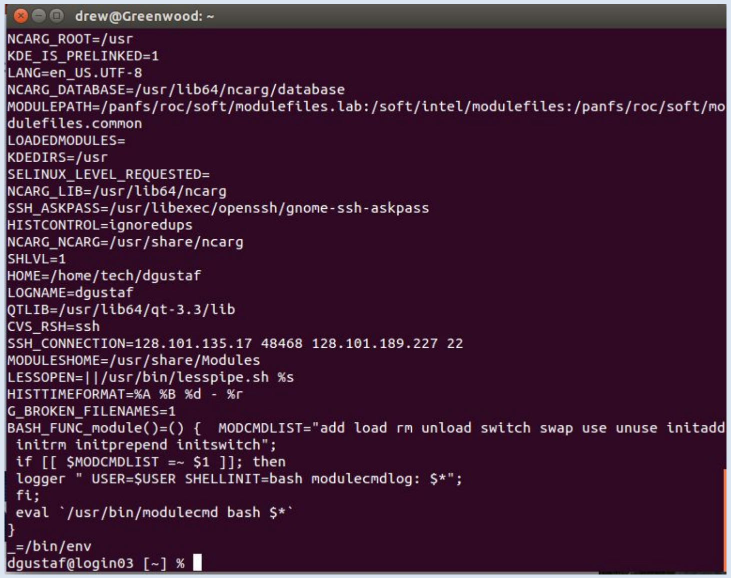
- There are now many flavors of Linux, called “distributions”.
- MSI uses a distribution called CentOS.
- Most distributions now have a Windows-like graphical user interface (GUI).
- The primary way to interact with Linux is still the command line using a terminal.

UNIVERSITY OF MINNESOTA

Minnesota Supercomputing Institute

The Terminal

The primary way to interact with Linux is the terminal.



```
NCARG_ROOT=/usr
KDE_IS_PRELINKED=1
LANG=en_US.UTF-8
NCARG_DATABASE=/usr/lib64/ncarg/database
MODULEPATH=/panfs/roc/soft/modulefiles.lab:/soft/intel/modulefiles:/panfs/roc/soft/mo
dulefiles.common
LOADEDMODULES=
KDEDIRS=/usr
SELINUX_LEVEL_REQUESTED=
NCARG_LIB=/usr/lib64/ncarg
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
NCARG_NCARG=/usr/share/ncarg
SHLVL=1
HOME=/home/tech/dgustaf
LOGNAME=dgustaf
QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
SSH_CONNECTION=128.101.135.17 48468 128.101.189.227 22
MODULESHOME=/usr/share/Modules
LESSOPEN=||/usr/bin/lesspipe.sh %s
HISTTIMEFORMAT=%A %B %d - %r
G_BROKEN_FILENAMES=1
BASH_FUNC_module()=() {  MODCMDLIST="add load rm unload switch swap use unuse initadd
 initrm initprepend initswitch";
 if [[ $MODCMDLIST =~ $1 ]]; then
 logger " USER=$USER SHELLINIT=bash modulecmdlog: $*";
 fi;
 eval `/usr/bin/modulecmd bash $*`
}
_=/bin/env
dgustaf@login03 [~] %
```

<< The terminal is a command line interface. It accepts text commands.

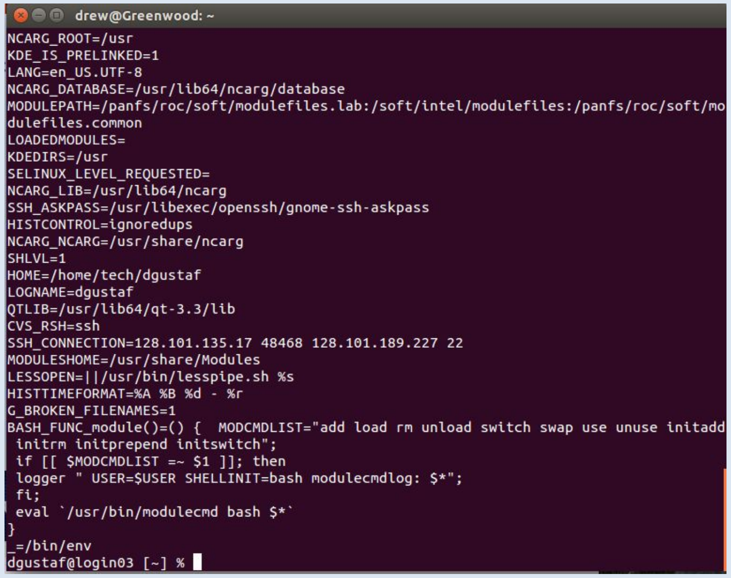
- At first, using the terminal feels like you “can’t see”. Eventually, you feel like you “can’t see” without it.
- The terminal is fast.
- To open a terminal in CentOS, right-click and select “Open in terminal”.

UNIVERSITY OF MINNESOTA

Minnesota Supercomputing Institute

The Terminal

The primary way to interact with Linux is the terminal.



```
NCARG_ROOT=/usr
KDE_IS_PRELINKED=1
LANG=en_US.UTF-8
NCARG_DATABASE=/usr/lib64/ncarg/database
MODULEPATH=/panfs/roc/soft/modulefiles.lab:/soft/intel/modulefiles:/panfs/roc/soft/mo
dulefiles.common
LOADEDMODULES=
KDEDIRS=/usr
SELINUX_LEVEL_REQUESTED=
NCARG_LIB=/usr/lib64/ncarg
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HISTCONTROL=ignoredups
NCARG_NCARG=/usr/share/ncarg
SHLVL=1
HOME=/home/tech/dgustaf
LOGNAME=dgustaf
QTLIB=/usr/lib64/qt-3.3/lib
CVS_RSH=ssh
SSH_CONNECTION=128.101.135.17 48468 128.101.189.227 22
MODULESHOME=/usr/share/Modules
LESSOPEN=||/usr/bin/lesspipe.sh %s
HISTTIMEFORMAT=%A %B %d - %r
G_BROKEN_FILENAMES=1
BASH_FUNC_module()=() {  MODCMDLIST="add load rm unload switch swap use unuse initadd
 initrm initprepend initswitch";
 if [[ $MODCMDLIST =~ $1 ]]; then
 logger " USER=$USER SHELLINIT=bash modulecmdlog: $*";
 fi;
 eval `/usr/bin/modulecmd bash $*`
}
_=/bin/env
dgustaf@login03 [~] %
```

- The terminal uses a language interpreter, or “shell”, called Bash.
- The terminal acts as if you are standing in one particular directory, the “working directory”.
- When you first open the terminal, you are standing in your “home directory”.
- The terminal generally executes commands without asking for confirmation.

Basic Commands

Most commands are rough abbreviations of English phrases.

- List directory contents: `ls`
- Print working directory: `pwd`

Directories are like folders, they can be nested inside each other. The / symbol separates nested directories, or files and directories.

Example: `/home/tech/dgustaf/document.txt`

File document.txt is inside directory dgustaf, inside directory tech, inside directory home.

The list of directories leading up to a file is called the "path" to the file.

Tab Completion

One feature that provides a great speed boost is tab completion. If you press the tab key, the terminal will try to guess what you are about to type, and will try to “complete” your command. Give it a try.

Command History

The terminal keeps a history of your most recent commands. Use the up and down keys to cycle through your most recent commands. This can be helpful if you want to reuse a previous command.

Basic Commands

List directory contents:	`ls`
Print working directory:	`pwd`
Change directory:	`cd`
Make directory:	`mkdir`
Remove file or directory: Careful! It doesn't ask to confirm.	`rm`
Move file or directory:	`mv`
Copy a file or directory:	`cp`

Special symbols used with commands:
A single dot . stands for the current directory.
A double dot .. stands for one directory higher.
The tilde ~ stands for the home directory.

Interrupt a Command

If you wish to interrupt and stop what the terminal is doing, press the keys:

Ctl C

This can be very helpful when you need to suddenly stop a program or command.

Interrupting a command or program may cause loss of data being operated on, and the interrupted command may have partially completed in an unpredictable manner.

Command Options

Most commands accept options that make them act slightly differently. The options usually begin with the minus sign - and are called “flags”.

Example:
To remove directory “test”, and all its subdirectories and files, use the command:

```
rm -rf test
```

Use the `man` command to view other command “manual pages”, which contain information on options.

Example:
To view the options for the `rm` command use:

```
man rm
```

Wildcard: *

The symbol * is the “wildcard”, which matches everything. This is useful when operating on groups of files or directories.

Example:
List the contents of all directories with names starting with letter p:

```
ls p*
```

Example:
Remove all files with names ending in txt:

```
rm *txt
```

Be careful! Wildcards can be dangerous!

Editing Text Files

To edit a text file from the terminal, use a text editor such as nano.

Example:
Create a text file named newfile.txt, and edit it:

```
nano newfile.txt
```

In nano **Ctl O** will save the file, and **Ctl X** will exit.

Another good editor is `vi`. The `vi` editor has more complicated commands, but is fast after you learn it.

Reading Text Without Editing

To read a text file from the terminal, without editing it, use a command such as: `less`

Example:
Read a text file without editing it:

```
less newfile.txt
```

Directing Output

The output of a command can be directed to a file using the > symbol.

Example:
Direct the output of `ls` to a file:

```
ls > list.txt
```

Then read the file with: `less list.txt`

The > symbol overwrites the previous file. To append to a file use >>

Example:
Append the working directory to the end of file list.txt:

```
pwd >> list.txt
```

Directing Output

The output of one command can be sent directly into another command using the pipe symbol: |

Example:
Direct the output of `ls` into the `less` program for reading:

```
ls | less
```

UNIVERSITY OF MINNESOTA

Minnesota Supercomputing Institute

Remote Access

To access another remote Linux system, use the Secure Shell command: `ssh`

Example:
Access the MSI login machine:

```
ssh login.msi.umn.edu
```

OR

```
ssh username@login.msi.umn.edu
```

Example:
Access the MSI lab server cluster:

```
ssh login.msi.umn.edu
```

And then from there:

```
ssh lab.msi.umn.edu
```

Type `exit` to close the connection.

Remote Access with Graphics

You can obtain remote access with graphics using: `ssh -X`

Example:

```
ssh -X login.msi.umn.edu
xeyes
```

A better way to handle intensive remote graphics is our NICE system.
See: https://www.msi.umn.edu/content/nice

Remote File Copy

To copy a file to or from a remote Linux system, use the Secure Copy command: `scp`

Example:
Copy a file to the MSI login machine:

```
scp localfile.txt login.msi.umn.edu:~/
```

Example:
Copy a file from the MSI login machine:

```
scp login.msi.umn.edu:~/remotefile.txt .
```

Example:
Copy all files ending in .txt to the MSI login machine:

```
scp *.txt login.msi.umn.edu:~/
```

File Permissions

In Linux, every file and directory belongs to a user and a group. Files and directories have a set of permissions controlling who can **read**, **write**, or **execute** them.

To see the files in the current directory along with their permissions, use the command: `ls -l`

Example `ls -l` output:

```
drwxrwxr-x  2 dgustaf tech 4096 Sep 11 14:20 2D_Integration
-rw-------  1 dgustaf tech   11 Sep 15 15:45 newfile.txt
```

The first position indicates directory or file.
The next positions are in groups of 3 corresponding to
user, **group**, **other**.

File Permissions

Changing Permissions:

To change file/directory permissions use the command: `chmod`

Examples:

Give a file read and write permissions for group members:

```
chmod g+rw file.txt
```

Remove execution permissions for other (non-group) users:

```
chmod o-x file.txt
```

Give read permissions for all users (user, group, other):

```
chmod a+r file.txt
```

Give execution permissions for the user (owner) and group:

```
chmod ug+x file.txt
```

File Permissions

Changing Ownership:

To change file/directory ownership use the command: chown

Example:

Change a file ownership to user1, group1:

```
chown user1:group1 file.txt
```

Executables

Files with execute permission are usually programs. These can be “executed” by typing the name of the file including the path.

Example:
Execute program.exe in the current working directory:

```
./program.exe
```

Example:
Execute program.exe located in /some/directory:

```
/some/directory/program.exe
```

There are some places the system automatically checks for programs, such as the /bin directory. These programs can just be executed by name. Most commands are actually tiny programs.

Environmental Variables

You can store a value in a variable using a command like:

```
VARIABLE=something
```

You can see the value stored in the variable using the echo command like:

```
echo $VARIABLE
```

Certain variables are special. The variable PATH controls which additional places the system will check for programs.
The variable HOME lists the location of your home directory.

To make a variable accessible to sub-processes use `export`:

```
export VARIABLE=something
```

Linux from OS X or Windows

Apple OS X is Unix-like operating system, similar to Linux. You can open a terminal within OS X, and most commands are similar to Linux. You can connect to MSI systems using ssh.

From Windows, some utilities are available to connect to Linux systems, or mimic Linux.

To connect to a remote Linux system from Windows, a good utility is PuTTy: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

To mimic (emulate) Linux within Windows, a good utility is Cygwin: https://www.cygwin.com/

Software Modules

Modules are used to alter environmental variables, in order to make software available. MSI has hundreds of software modules.

Module Commands:

Description	Command	Example
See all available modules:	`module avail`	`module avail`
Load a module:	`module load`	`module load matlab/2015a`
Unload a module:	`module unload`	`module unload matlab/2015a`
Unload all modules:	`module purge`	`module purge`
See what a module does:	`module show`	`module show matlab/2015a`

UNIVERSITY OF MINNESOTA

Minnesota Supercomputing Institute

Scripts

It is possible to store a series of commands in a text file, and then execute the text file. Such a file is called a "script". A script is really a small program.

Example:
Make a script that prints "Hello World!"
Edit a file named hello.sh to contain:

```
#!/bin/bash -l
STRING="Hello World!"
echo $STRING
```

Give hello.sh excution permissions:

```
chmod u+x hello.sh
```

Execute the script:

```
./hello.sh
```

More Commands

Description	Command	Example
Find files or directories:	`find`	`find . -name file.txt`
Read file into terminal:	`cat`	`cat file.txt`
Search files for string:	`grep`	`grep hello *`
Ping remote computer:	`ping`	`ping www.google.com`
Count words in a file:	`wc`	`wc file.txt`
See top running programs:	`top`	`top`
List processes:	`ps`	`ps aux`
Kill a process:	`kill`	`kill 12345`
Clear the terminal:	`clear`	`clear`

More Commands

Description	Command	Example
Print date and time:	`date`	`date`
Extract/Create archive:	`tar`	`tar -xf archive.tar.gz`
Print program location:	`which`	`which ls`
Print member groups:	`groups`	`groups`
Print command history:	`history`	`history`
Print first part of file:	`head`	`head -n 5 file.txt`
Print last part of file	`tail`	`tail -n 5 file.txt`
List logged in users:	`who`	`who`
Print system info:	`uname`	`uname -a`

More Examples

Change to the directory you were just previously in:

```
cd -
```

Find and delete all files with names beginning with ABC:

```
find . -type f -name ‘ABC*’ -delete
```

Search the list of processes for lines including dgustaf:

```
ps aux | grep dgustaf
```

List all files including hidden files (names begin with dot):

```
ls -a
```

Start a program and make it run in the background:

```
./program.exe &
```

Git

Git is a program for doing file syncing and tracking file changes. It is very useful to keep files synced between systems, and to track what changes are made. It is highly used for code collaboration

The university has a local Git server: github.umn.edu

Description	Command
Clone git repository:	`git clone`
Print repository status:	`git status`
Add changed file:	`git add file.txt`
Add all changes	`git add -A`
Commit repository:	`git commit -m 'Message'`
Push changes to remote server:	`git push`
Pull changes from remote server:	`git pull`

Minnesota Supercomputing Institute

Web: www.msi.umn.edu

Email: help@msi.umn.edu

Telephone: (612) 626-0802

UNIVERSITY OF MINNESOTA