

# HPC Fundamentals Introduction to Unix



**Robert Whitten Jr**



U.S. DEPARTMENT OF  
**ENERGY**

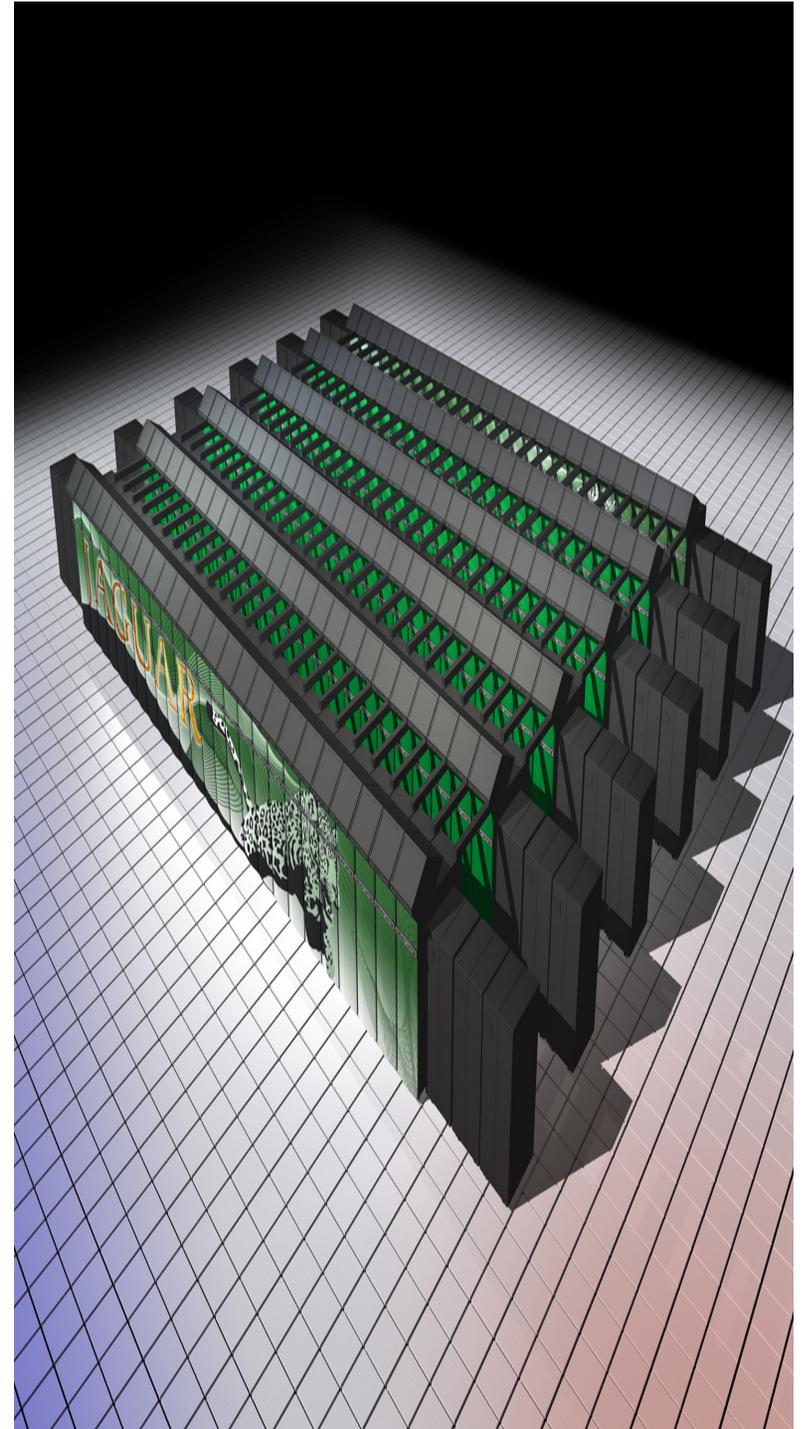


**OAK RIDGE NATIONAL LABORATORY**

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

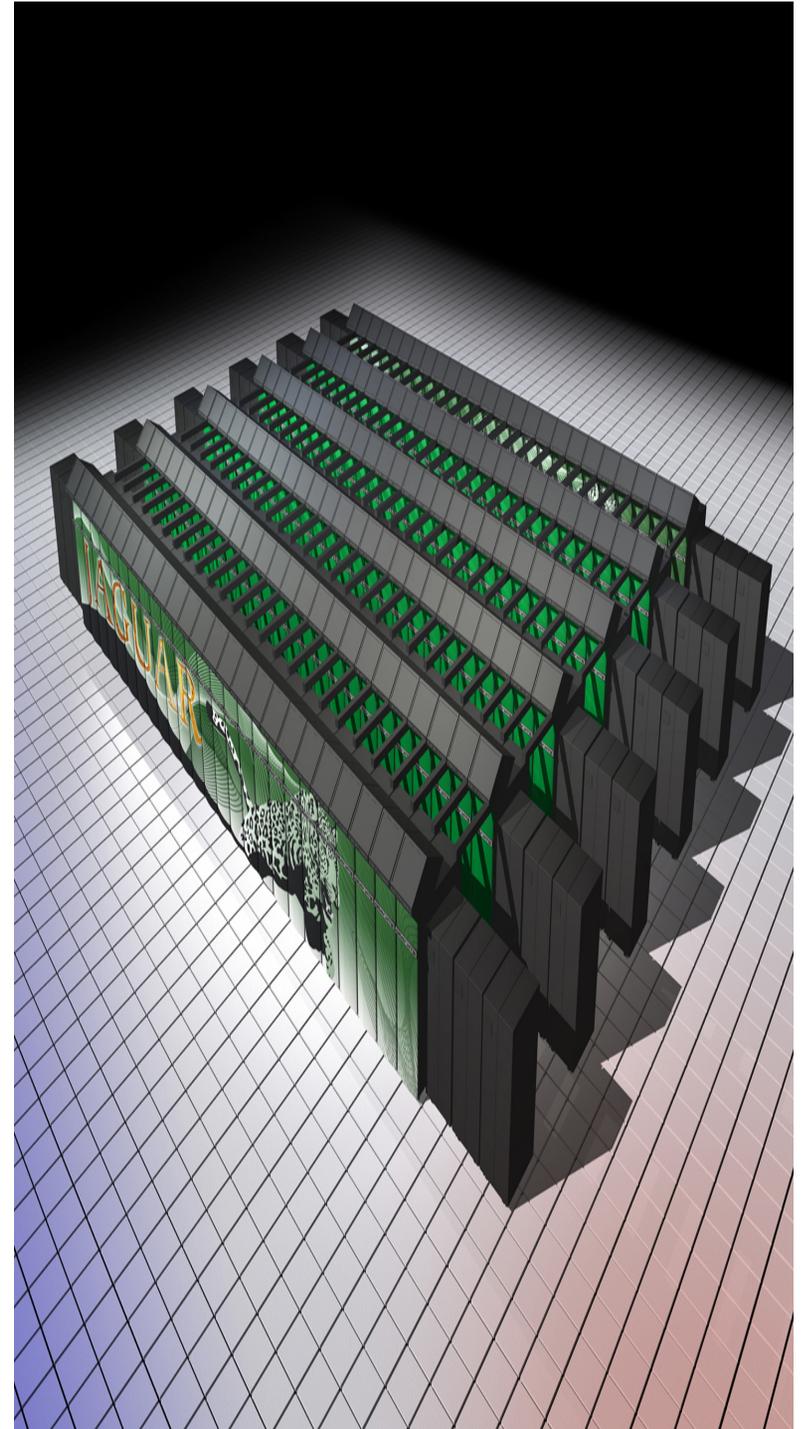
# Welcome!

- Today's Agenda:
  - Questions from last week
  - The basics
  - What is Unix?
  - Unix commands you can't live without



# The basics

- Hardware
- Software
- Application software
- System software



# Hardware

- Everything that can be touched in a computer
- Microprocessor
- Primary memory
- Secondary memory
- Network cables
- Printers
- Keyboards, mice, etc



# Software

- Programming used to be difficult
  - Rewire the whole machine each time
- Makes the hardware usable
- System software (operating systems)
  - Controls hardware
  - Applications do not need to know how to use hardware
  - Kernel vs. utilities
  - GUI vs. CLI
- Application software
  - Word processing
  - Spreadsheets
  - Games

# Unix

- Operating system
- Developed in early '70s by AT&T at Bell Labs
- Multi user system
- Unix has come to mean any Unix-like operating system
- Andrew Tanenbaum created Minix
  - Textbook demonstration
- Linus Torvalds created Unix-like kernel
  - Linux was born
  - Technically linux is the kernel only

# Unix characteristics

- Multi-user operation
  - Accounts for users
  - Permissions based
- Command line interface (CLI)
  - No GUI (sort of)
    - X11 is a standard for doing GUI on unix systems
- Utility programs
  - Navigate system
  - Execute programs
- Device management through files

# Unix characteristics

- Data security
  - Permissions based
    - Read / write / execute
  - File system based
- Data processing through filters
  - Text manipulating programs used heavily

# Unix – accounts

- All users have a distinct account name
  - i.e. bob, mary, userx11, superdude, etc
- All accounts have passwords
  - Don't use common passwords (name, birthday, 'password', etc)
  - Authentication vs. authorization
- All accounts have a default home directory
  - More on file systems and directories in a bit
- All accounts have configuration files
  - For individual preferences
- All accounts have a command interpreter

9 □□□ Program that accepts and executes commands

# Unix - permissions

- All data is stored in files
  - Files are collections of data lumped together
    - Addresses, recipes, raw data, etc
- All files have permissions
  - Read / write / execute
- Permissions based on:
  - Who you are & what group you are in
- Permissions are divided into three categories

# Unix - permissions

- Example:

```
-rw-r--r-- 1 csep100 ccsstaff 303 Aug 28 14:21 staff 222 Jun 4 mpi.pbs
```

- Close up:

**-rw-r--r--**

# Unix commands you can't live without

- man – read system manual pages
- pwd – identify the working directory
- cd – change the working directory
- echo – display a string
- ls – display contents of directory
- cat – display text of file
- more – display text of file
- cp – copy a file
- mv – move a file
- rm – remove a file
- mkdir – create a directory
- rmdir – remove a directory
- exit – end session

# Unix commands you can live without, but who'd want to

- ssh – initiate remote connection
- scp – copy a remote file
- tar – archive files
- find – find files or directories
- less – less is more (with benefits)
- vi – edit files
- env – environment variables
- hostname – list name of current machine
- ln – create links
- history – show command history
- ps – show running processes
- kill – kill a running process
- chmod – change permissions

# Unix file system

- Hierarchical file structure
  - Directories
  - Files
- Directories
  - Contain files and/or other directories (subdirectories)
- Files
  - Contain data (text, binary, etc)

# Hierarchical (tree) file system

- There is one parent directory per file system
  - Root aka ‘/’
  - Everything else is contained in this directory
    - Subdirectories
    - Files
- Pathnames – name (address) of the file/directory
  - /ccs/home/csep100
  - Absolute vs. relative pathnames
- Navigating directories
- Creating files, directories
  - Redirection, pipes

# Shell scripts

- Programs to help automate recurring task
- Text files that are interpreted by shell program
  - Interpreted vs compiled languages
- Example

# Shell script basics

- Comments
- Values
- Variables
- Arrays
- Selection
- Loops

# Shell script - comments

- Comments are not executed
- Useful in documenting you scripts
- # = comment everything to right
- Exception is very first line of script
  - #!/bin/bash

# Shell script - values

- Values are object used in your script
  - String values – ‘bobby’
  - Numerical values – 94
- String values are quoted
  - ‘bobby’ – just a string, no processing
  - “bobby” – a string, but with processing
  - `bobby` - execute this string and get output back
- Numerical values can have math performed on them using the special form  $\$( ( ))$ 
  - $x=\$(2+2)$
  - echo \$x
  - 4

# Shell script - variables

- Use variables when the values may change
- Example
  - x=hello
  - echo \$x
  - hello
  - x=5
  - echo \$x
  - 5
- If you want to treat a string as a variable
  - let x=5\*5

# Shell scripts - arrays

- Arrays are variable with 1 or more elements
  - `x=(a b c d)`
  - `echo ${x[0]}`
  - a
  - `echo $x`
  - a
  - Where's everything else?
  - `echo ${x[@]}`
  - a b c d

# Shell scripts - selection

- If this, then that, else something else

- if ... then ... elif ... fi

```
#!/bin/bash
if [ $1 = 'hello' ];
then
    echo hello yourself
elif [ $1 = 'howdy' ];
then
    echo howdy to you too
else
    echo nobody home
fi
```

# Shell script - loops

- For this number of times, do something

```
for name in bobby doug frank; do
    echo $name
done
```

```
let x=1
until [ $x -eq 10 ]; do
    echo $x
    let x=$x+1
done
```

```
let x=1
while [ $x -ne 10 ]; do
    echo $x
    let x=$x+1
done
```

# Exercise – Part 1

- Create a bash script that performs the following actions:
  - Create a text file called ex1.txt
  - Overwrite the ex1.txt and add to it your: name, email
  - Create a directory called exercise1
  - Create four subdirectories within the exercise1 directory: sub0/, sub1/, sub2/, sub3/
  - Create 1 subdirectory in sub0, 2 in sub1, 3 in sub2, 4 in sub3

# Exercise – Part 2

- Modify your shell script to do the following:
  - Copy ex1.txt using relative path names into the following subdirectories:
    - exercise1/sub0/sub0
    - exercise1/sub2/sub1
    - exercise1/sub3/sub0
    - exercise1/sub3/sub1
  - Rename the files just copied to <your name>.txt
  - For each file just renamed, copy it back up one directory:
    - Example: exercise1/sub3/sub1/bobby.txt -> homework1/sub3/bobby.txt

# Questions?



<http://www.olcf.ornl.gov>