Getting Into the System

Mobeen Ludin

What is an Operating System?

- The operating system (OS) is the program which starts up when you turn on your computer and runs underneath all other programs - without it nothing would happen at all.
- **PLEASE STOP CONFUSING ME!!! Say it in SIMPLE TERMS!!!**
- In simple terms, an operating system is a *manager*. It manages all the available resources on a computer, from the CPU, to memory, to hard disk accesses.
- Tasks the operating system must perform:
  - **Control Hardware** - The operating system controls all the parts of the computer and attempts to get everything working together.
  - **Run Applications** - Another job the OS does is run application software. This would include word processors, web browsers, games, etc...
  - **Manage Data and Files** - The OS makes it easy for you to organize your computer. Through the OS you are able to do a number of things to data, including copy, move, delete, and rename it. This makes it much easier to find and organize what you have.

Unix History:

- The UNIX operating system was born in the late 1960s. It originally began as a one man project led by Ken Thompson of Bell Labs, and has since grown to become the most widely used operating system.

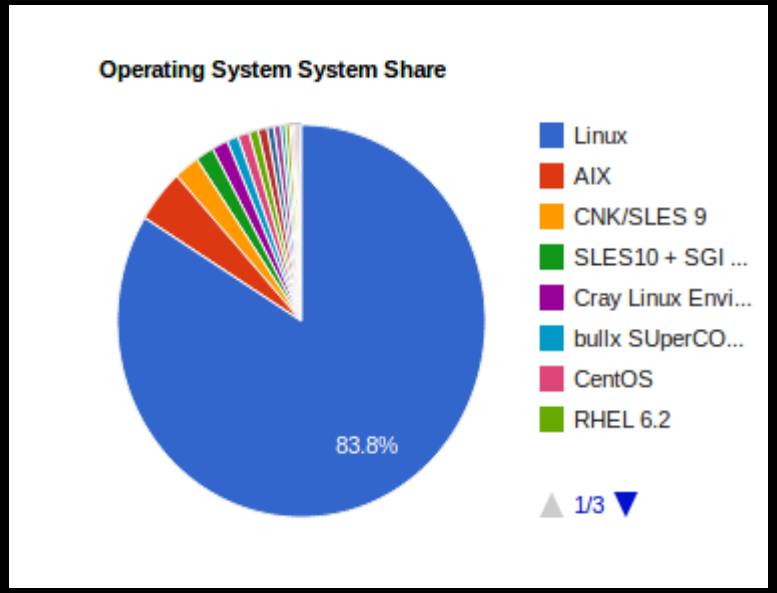
- In the time since UNIX was first developed, it has gone through many different generations and even mutations.
  - Some differ substantially from the original version, like Berkeley Software Distribution (BSD) or Linux.
  - Others, still contain major portions that are based on the original source code.

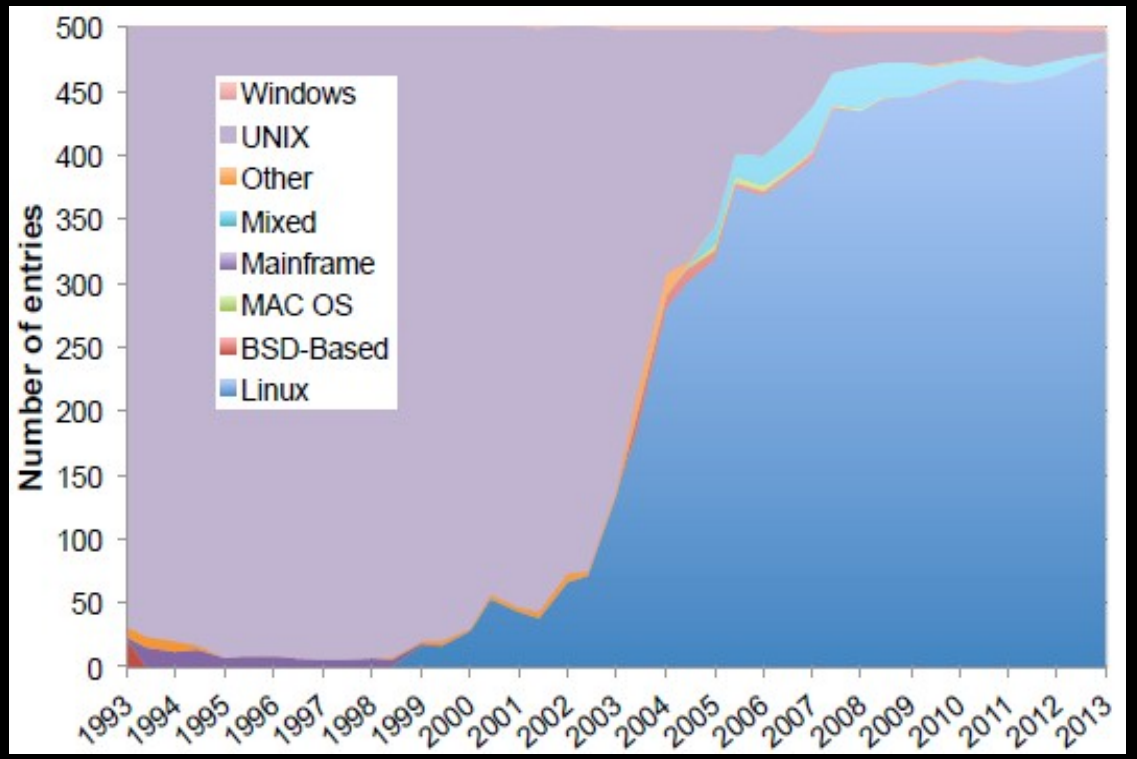
- An interesting and rather up-to-date timeline of these variations of UNIX can be found at http://www.levenez.com/unix/history.html.

Parts of the Unix/Linux OS:

- **The Kernel** - handles memory management, input and output requests, and program scheduling. Technically speaking, the kernel is the OS. It provides the basic software connection to the hardware. The kernel is very complex and deals with the inner workings of these things.
- **The Shell and Graphical User Interfaces (GUIs)** - basic UNIX shells provides a “command line” interface which allows the user to type in commands. These commands are translated by the shell into something the kernel can comprehend, and then executed by the kernel.
- **The Built-in System Utilities** - are programs that allow a user to perform tasks which involve complex actions. Utilities provide user interface functions that are basic to an operating system, but which are too complex to be built into the shell. Examples of utilities are programs that let us see the contents of a directory, move & copy files, remove files, etc...
- **Application Software & Utilities** – these are not part of the operating system, per se. They are additional programs that are bundled with the OS distribution, or available separately. These can range from additional or different versions of basic utilities, to full scale commercial applications.

OS used on Supercomputers:





Source: Linux Foundation Report: Linux dominates 90% of systems.
http://www.linuxfoundation.org/publications/linux-foundation/top500_report_2013

What are the Different Nodes on BW?

A Node is a single computing device on network. Supercomputers usually have at least three type of Nodes:

- *Login nodes/Service nodes:* The node you access when you first log in to the system. Login nodes offer the full Linux environment operating system, are used for basic development tasks such as editing files and compiling code, generally have access to the network file system, and are shared resources that may be used concurrently by multiple users.
- *Compute nodes:* The nodes on which production jobs are executed. Compute nodes run a lightweight operating system called `Compute Node Linux (CNL)`, can be accessed only by submitting jobs through the batch management system (typically `PBS Pro`, `Moab/TORQUE`, or `Platform LSF`), generally have access only to the high-performance parallel file system (typically `Lustre` or `Panasas`), and are dedicated resources, exclusively yours for the duration of the batch reservation.
- *Administrative nodes:* Managing network, storage, backups, etc... resources. Users shouldn't worry about these nodes unless they are administrators.

How do you login to a Supercomputer

Remote login:

Once upon a time, people thought they had a need to be able to "easily" access other machines without going through a normal login

Out of this desire, the "**r-commands**" were born

- **`rcp`** – remote file copy
- **`rlogin`** – remote login
- **`rsh`**, remsh, remote_shell – variants of remote shell
- **`rwho`** – who is logged on other systems on your net

Because of the security issues, many systems no longer allow *rcp, rlogin, or rsh*

Instead, use something like **`SSH`**, the secure shell as replacement for r-commands family.

- **SSH** is a protocol for secure remote access to a machine over untrusted networks.

SSH Syntax:

Syntax: `ssh [-X] username@remote.host`

- *username* – username you want to use on the remote machine, *ssh* defaults to your login id on the local machine
- *remote.host* – name of the remote machine if on the local network or fully qualified internet name if on a remote network
  - `bwbay`
  - `bwbay.ncsa.illinois.edu`
- You can also provide the username with the remote.host name
  - `$ ssh -X instr004@bwbay.ncsa.illinois.edu`
- **Exercise 1: Connecting to BW [15 minutes]**
- **Exercise 2: Common Linux Commands [30 minutes]**
  - `ls, cp, mv, pwd, cat, w, who, quota, less, more, touch, rm, mkdir, module, make`
  - `$ man wget` (to learn more about a command from manual page)

Copying files to/from Blue Waters:

Syntax: `scp source_file destination_file`

- source_file and destination_file can either or both use the full user and system name like in ssh
  - `$ scp mludin@login.shodor.org:my_file .`

★ **Exercise 3: Transferring Files To/From Blue Waters [15 minutes]**

- `scp` (remote copy)
- `cp` (local copy)
- `wget` (url copy)

```
$ wget http://shodor.org/~mludin/BW_Institute.tar
```

Utilities:

Compilers & Linkers:

- **`PrgEnv-cray`** (Cray Compiler suite)
- **`PrgEnv-pgi`** (PGI compiler suite)
- **`PrgEnv-gnu`** (GNU compiler suite)

Compiler Wrapper Scripts:

- C = **`cc`**
- C++ = **`CC`**
- Fortran = **`ftn`**

Accelerator Programming Environments (for XK nodes):

- **`OpenACC`**
- **`NVIDIA CUDA`**

Debuggers:

- C/C++ debuggers: **`gdb`**
  - DDT (Distributed D. T.)

Interpreters:

- Perl: **`perl`**
- Python: **`python, python2.x`**

Text Editors:

- **`vim/vi`**
  - **`vimtutor`** (vim step-by-step tutorial)
- **`emacs`** [can be opened with X11]

Managing User Environment:

- Modules

Editors on BW

- Using Vim/vi:
  - `$ vim file_name.extension`
    - read (command) mode
    - write (insert or text) mode
    - save, exit/quit [esc (`:q`, `:q!`, `:w`, `:wq`, etc)]
- Using emacs:
  - `$ emacs file_name.extension`
    - move around (keys)
    - edit mode
    - exit mode
      - exit with saving/without saving
  - `emacs_x11` (graphical emacs with `-X`/`-Y` on SSH)

Types of Jobs on Blue Waters

Two type of jobs – **interactive** and **batch**

- Interactive mode for debug and optimization
- Batch mode for normal job runs

Steps for setting up an interactive job:

- Determine the resources needed
- Pick a queue that will provide the required resources
- Submit the job using aprun command

Example:

- Interactive jobs:
  - `$ qsub -I -l nodes=1`
  - `$ qsub -I -l nodes=2:ppn=32:xe -l walltime=00:30:00`

Rest of the day we will be using Interactive session only.

Using Modules on Blue Waters:

What the heck is module?

Module is a software stack used on many supercomputers to easily manage user development/programming environment. When using modules, users are not required to specify explicit paths for different executables, libraries, compilers versions and other environment variables.

For Example:

Switching between the version of compilers, or other scientific utilities such as **FFTW** you have to make appropriate changes to your makefiles. But when using modules, all is automated for you. No need for paths in the makefile.

Lets try it:

```
$ module list
$ module swap PrgEnv-cray PrgEnv-gnu
$ module list
```

Compiling on Blue Waters:

What is a compiler?

Everything on Blue Waters is controlled by **`modules`**

- Remember BW development environment uses COMPILER WRAPPERS
  - **`cc`** (C)
  - **`CC`** (C++)
  - **`ftn`** (Fortran)

NOTE: **DO NOT** use **`gcc`**, **`g++`**, **`f90`**, **`mpicc`**, etc..Also make sure Makefiles are correct.

- $ **`cd Hello_world/`**
- $ **`less Makefile`** (use up/down arrows to read the file and q to quit)

Switch default (Cray) compiler suites to GNU using **`modules`**:

$ **`module swap PrgEnv-cray \ PrgEnv-gnu`**

- Compile your first code using **`make`**:
  - $ **`ls -al`**
  - $ **`make build-ll`**
  - $ **`ls -al`** (what files were created by make??)

Running Jobs Overview

Researcher

Login

Command Line

When you connect to a resource, you are on a login node shared by many users.

Batch Script

Commands for code execution, copy input files to scratch,... Specify number/type of nodes, length of run, output directory, ...

Data

Job

*Store project files such as source code, scripts, and **small** input data sets to your Home directory.*

Login Nodes

Use for tasks such as file editing, code compilation, data backup, and job submission.

Run jobs by submitting your batch script to the compute nodes using the "qsub" command.

Home

Scratch

File System

Data

Read/write data from compute nodes to your Scratch directory.

Compute Nodes

Your job is submitted to a queue and will wait in line until nodes are available. Queues are managed by a job scheduler that enables jobs to run efficiently.

Source: https://bluewaters.ncsa.illinois.edu/running-your-jobs

Running Jobs on Blue Waters:

Jobs on BW are managed through the use of:

- Resource manager: TORQUE
- Workload manager: Moab
- Application Launcher: Application Level Placement Scheduler (ALPS)

Commands for managing jobs on Blue Waters are a subset of **`PBS`** (Portable Batch System) commands.

Application launcher (**`aprun`**) utility launches applications on compute nodes. ALPS handles application placement and execution.

Submitting Jobs on Blue Waters

Steps for setting up a batch job:

- Submit job:
  - `$ qsub my_script.pbs`
- PBS Script:
  - Sample scripts are at `/sw/userdoc/samplescripts`
  - Specify resources needed
  - Provide file names for stdout and stderr
  - Define environmental variables
  - Load needed modules
  - Launch the job via the aprun command

PBS Script - Resources and Notification

```
#PBS -l nodes=2048:ppn=32:xe
#PBS -l walltime=01:20:00
#PBS -N testjob
#PBS -e $PBS_JOBID.err
#PBS -o $PBS_JOBID.out
#PBS -m bea
#PBS -M username@host
cd $PBS_O_WORKDIR
module load craype-hugepages2M
export OMP_NUM_THREAD = 2 ## export your_environment_variables

aprun -n 65536 ./app_executable < in >out.$PBS_JOBID
```

Scientific Computing/Modeling

- Lets have some FUN!!!
  - Run Hello-World
  - Run GalaxSee
  - Run Life
  - Run Pandemic
  - Run Area_Under_Curve

NOTE: Do NOT forget to Observe!!

```
$ time aprun -n 10 ./GalaxSee.cxx-mpi 1000 100 1000 0
```

Exit interactive session: by entering **`exit`** command

★ **Exercise 4: Running First Program on Blue Waters [30 minutes]**