UNIX System Programming
Lecture 1: Introduction

Outline
What is UNIX?
What is Linux?
UNIX Programming Environment
Compiling C and C++ Programs
Reference
BLP: Chapter 1

Lecture 1: Introduction
What is UNIX? Features

- A _________, multitasking operating system.
- UNIX is portable (written in C).
- UNIX is secure (file and process security).
- A UNIX system includes 100s of utilities, tools and libraries. Tools for text processing and programming are standard.

Lecture 1: Introduction
What is UNIX? Features

- The TCP/IP protocols (Internet) were developed under UNIX and are now a core part of the OS.
- While not a core part of the OS, a Graphical User Interface is standard (X Window System or X).
- The _________ (portable operating system interface) standards are now used to define UNIX-like systems. Standards define interfaces, shells, tools, security, etc.

Lecture 1: Introduction
What is UNIX? History

- UNIX was developed by Bell Labs (AT&T) in the early 1970s. AT&T gave the source code of the OS to universities and licensed it to many other companies. (Several companies developed their own versions of UNIX.)
- At Berkeley, several enhancements resulted in Berkeley System Derived (______) UNIX.
- The UNIX trademark belongs to the Open Group. SCO owns the original AT&T source.

Lecture 1: Introduction
What is Linux? History

- Linux is a free, POSIX compatible OS originally written by Linus Torvalds. It was first released in 1991. (Linus was a student in Finland at the time.) It is now maintained and updated by people around the world.
- The term Linux properly refers to only the _________ of the OS, but is usually used to describe the complete OS (kernel, libraries, core utilities and video support) or distribution.
Lecture 1: Introduction
What is Linux? Features

- Like most modern OSes Linux supports: true multitasking, ________, virtual memory, shared libraries, demand loading, shared copy-on-write executables, memory management, loadable device driver modules, video frame buffering, and TCP/IP networking.
- Although originally written for the Intel 80386 processor it has been ported to over 20 other processors.

Lecture 1: Introduction
What is Linux? Distributions

- There are several Linux distributions: Ubuntu, Fedora, Mint, Debian, Slackware, ArchLinux, PCLinuxOS.
- A distribution consists of the complete Linux OS, plus administration and user applications. An installation program (___________) is also included.
- Distributions provide software in packages along with a package management program.

Lecture 1: Introduction
What is Linux? The Mascot

_____ the Linux Penguin

Lecture 1: Introduction
UNIX Prog. Environment

- Cygwin
  - Cygwin is a free UNIX-like environment for ___________. It includes C and C++ compilers, shells, Perl, Python, UNIX emulation libraries and X (and much, much more).
- MSYS/MinGW/MSYS2/MinGW-w64
  - MSYS is a minimal GNU system for Windows. It includes C and C++ compilers (MinGW), a shell and standard UNIX utilities (make, grep, etc). There is limited UNIX emulation support.
  - This is not a recommended setup for this course.

Lecture 1: Introduction
UNIX Prog. Environment

- CSLAB/csserver
  - CSLAB/KC267 dual-boot Linux and Windows.
  - Remote log on to csserver via ssh.
  - Putty and WinSCP are free ssh and sftp Windows clients.
  - ALL PROGRAMS MUST BE TESTED ON CSSERVER BEFORE SUBMISSION!!!!!!!!!
- VirtualBox Virtual Machine
  - Run Linux in a virtual machine.
  - Programming, ________, and administration
  - Recommended: Lubuntu/Ubuntu.

Lecture 1: Introduction
UNIX Prog. Environment

- Linux
  - Install Linux on your PC. Linux is not a Windows application. You will need to either dedicate a PC to Linux or set up a PC to dual-boot Linux and Windows.
  - If you have never installed Linux before get help from someone who has.
  - Most distribution installers can resize your Windows __________ to make room for Linux.
Lecture 1: Introduction
Compiling C and C++ Programs

- Create the source file using an editor:
  
  ```
  $ vi first.cpp
  ```
  
  ```
  #include <iostream>
  using namespace std;
  int main()
  {
    cout << "CS375 is off and running!";
    cout << endl;
    return 0;
  }
  ```

- Compile and run (or just “make first”)
  
  ```
  $ g++ -o first first.cpp
  $ ./first
  ```

- Refer to the man or info pages for info on g++

- Use the “-v” option to get full details:
  
  ```
  $ g++ -v -o first first.cpp
  Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
  Configured with: ...... 
  ```

Lecture 1: Introduction
The PATH Environment Variable

- The PATH variable contains a list of directories that is searched when looking for programs:
  
  ```
  $ echo $PATH
  /bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin
  ```

- A program not in the PATH can be run by using the complete pathname:
  
  ```
  $ ./first
  ```

- You can add the current directory to the PATH (not recommended):
  
  ```
  $ PATH=$PATH:
  $ first
  ```

Lecture 1: Introduction
Include Files

- Let's build a slightly more complex application:
  
  ```
  $ vi second.cpp
  ```

- Include utility routines in a header:
  
  ```
  #include "mymath.h"
  ```

- Include our prototypes:
  
  ```
  #include <iostream>
  ```

- To compile and link:
  
  ```
  $ g++ -o second -I. second.cpp mymath.cpp
  ```

- g++ automatically looks in the current directory so the “-I.” is optional here.

- Standard include directories /usr/include, /usr/local/include, /usr/include/c++/GCCVERSION are automatically searched.