Intro to Linux and C

CSCI 2400/ ECE 3217: Computer Architecture

Instructors:
David Ferry
Overview

- Linux
- C
- Hello program in C
- Compiling
History of Linux

- **Way back in the day: Bell Labs Unix**
  - Widely available to students and instructors
  - Very machine-independent

- **Some direct Unix branches (e.g. Berkeley Unix or BSD)**

- **Others were inspired by Unix**
  - Minix - by Andrew S. Tannenbaum, an educational micro kernel
  - Could re-implement the high-level design of Unix (e.g. Minix was originally system-call compatible with Unix)

- **Linus Torvalds saw Minix and wanted to do his own version**
  - Wrote basic kernel from scratch
  - Borrowed good ideas from Minix
  - Included early support for GNU project software
  - Completely free OS and system software
Linux Today

- Very small usage in desktop/laptop market (~3% in US)
- Android is the biggest OS in mobile computing (~53% of US)
  - Up to 85% of devices worldwide
- Linux drives internet servers (~97% of public servers)
- Linux drives supercomputing (~99% of TOP500 computers)
Getting Started with Linux at SLU

- Linux classroom and Linux lab on 1st floor Ritter
- Department Linux server: hopper.slu.edu
- Should use same username as SLU, but different password
- Talk to Dennis about password issues (office adjacent to lab)

- Recommended: Login to hopper.slu.edu via ssh
- Suggested: Work on local machine in lab
- Suggested: Login to hopper.slu.edu via NoMachine
- You may work however you like, but I can’t support other methods (e.g. Linux in a virtual machine on your laptop)
Logging in via SSH

- **From OSX – can use terminal directly**
  - Can transfer files with ‘scp’ command

- **From Windows – can use an ssh client**
  - My favorite: Secure Shell extension for Chrome browser
  - Plenty of others, just search for them
  - Transfer files via WinSCP

- **Username: your SLU username**
- **Hostname: hopper.slu.edu**

- **Via terminal:**
  
  ssh dferry@hopper.slu.edu
Using the command line

- Enter one command per line
- Lots of programs to accomplish what you want to do
  - Just search “How do I accomplish XYZ in Linux terminal?”

**Useful commands:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls</code></td>
<td>Lists contents of current directory</td>
</tr>
<tr>
<td><code>ls -l</code></td>
<td>Lists contents in list format</td>
</tr>
<tr>
<td><code>cd</code></td>
<td>Change current directory</td>
</tr>
<tr>
<td><code>mkdir</code></td>
<td>Make a new directory</td>
</tr>
<tr>
<td><code>rm</code></td>
<td>Remove a file</td>
</tr>
<tr>
<td><code>rm -r</code></td>
<td>Remove a directory</td>
</tr>
<tr>
<td><code>cp file1 file2</code></td>
<td>Copies <code>file1</code> to <code>file2</code></td>
</tr>
<tr>
<td><code>cat file</code></td>
<td>Prints <code>file</code> to the terminal</td>
</tr>
<tr>
<td><code>wget url</code></td>
<td>Downloads <code>url</code> to the current directory</td>
</tr>
</tbody>
</table>
Editing Text Files

■ Text files- very important!
  ▪ C programs for this class
  ▪ Very efficient storage for data and configuration

■ Classic editors: vi and emacs
  ▪ Hard to get started initially
  ▪ Way faster once you get the hang of it
  ▪ Designed for low-bandwidth, spotty connections (think phone modems)
  ▪ Definitely worth it

■ Other editors:
  ▪ Text editors- search them!
  ▪ GUI editors- search them!
For next time:

- Find a good Linux environment you’d like to use
- Try logging into hopper.slu.edu
- Next homework involves writing C code
Overview

- Linux
- C
- Hello program in C
- Compiling
The C Language

- Developed at Bell Labs to write Unix
  - Practical language for practical projects
  - Most OSes are written mostly in C (some assembly code)
  - Most system libraries and tools are written entirely in C

- Small, simple language
  - Easy to learn (especially at the time)
  - Easy to port to different platforms

- Designed to replace Assembly Language
  - Provides low-level access to memory
  - Most operations map closely to assembly language operations

- Strongly typed, static type checking
  - int, unsigned, float, double, char, etc.
  - No runtime protection
Type Checking in C

- Several primitive types:
  - int, unsigned, float, double, char, etc.

- The compiler *will not* warn about possibly unsafe operations:
  ```c
  int a;
  unsigned b;
  b = a;
  ```

- Correctness is up to the programmer!
  - This kind of stuff is usually a bad idea though...
Command Line Input in C

- The main() function has two arguments:
  - argc is the number of arguments
  - argv is a vector of strings that hold those arguments

E.g.: printing all values as strings

```c
int main ( int argc, char* argv[] ){
    for( i = 0; i < argc; i++ ){
        printf("%s\n", argv[i]);
    }
}
```
Converting Strings to Numbers

- How to convert “42” into the numeric value 42?

- **atoi()**
  - Fast, easy, but dirty
  - No safety or type checking
  - Undefined behavior on overflow

- **scanf()**
  - Works with floats and other data types
  - Undefined behavior on overflow

- **strtol()**
  - Robust error checking, industrial grade
Manual Pages! (A.K.A. man pages)

- All of C (and much more beside) is documented in the *manual pages*, use them!

- At the command prompt:
  
  - `man man` – manual for the manual pages()
  - `man atoi` – manual for the function atoi()
  - `man scanf` – manual page for the function scanf()

- Sometimes there are collisions:

  - `man printf` – manual page for the bash command printf
  - `man 3 printf` – manual page for the C function printf()
  - `man man` – shows man page sections
C Operators

- **Usual arithmetic operators:**
  - +, -, *, %, /, =

- **Bitwise operators:**
  - & - AND
  - | - OR
  - ^ - XOR
  - ~ - complement
  - << - left shift
  - >> - right shift

- **Ternary operator:**
  - (cond) ? (exec if true) : (exec if false)

- **Logical Operators:**
  - && - logical AND
  - || - logical OR
  - ! - logical NOT

- **Relational Operators**
  - == - True if equal
  - != - True if not equal
  - < - True if less than
  - > - True if greater than
  - <= - True if less or equal
  - >= - True if greater or equal
Hello, world! in C

#include <stdio.h>

int main( int argc, char* argv[] ){

    int var = 42;
    printf("Hello, world!\n");
    printf("Value of var: %d\n", var);

    return 0;
}

Compiling C programs

- We will use the gcc compiler

- If you have a program named prog.c:

  gcc -Wall -o prog prog.c

  -Wall turns on all warnings
  -o <name> output file name (default is a.out)
  Program files are listed by themselves
  Order isn’t important