Announcements

• Extra office hours:
  • Friday, August 7, 12:30-3pm (BA3289)
  • Monday, August 10, 1-4pm (BA3201)
• A4 is due tomorrow by 11:59pm
Exam
Evening of Tuesday, August 11

http://www.artsci.utoronto.ca/current/exams/
3 hours long

7 questions for a total of 80 marks
One double-sided 8.5x11 sheet of paper

No electronic aids
Course Evaluations

Please fill them out! All feedback is welcome and appreciated!
Agenda

• Big Picture Review
• Assignments
• Assorted Comments
• What’s next?
Ask About

- A topic
- A slide
- A code example
- A past exam question
- Suggested exercises
- Anything else...
Fun Statistics

• Code Commits

  • More than 4500 commits made across all student repositories

  • Average of 50 commits per student

  • More than ten students with over 100 commits

  • Highest committer with over 500 commits!
Fun Statistics

- Source Code
  - 3000 lines of assignment starter code & headers
  - Almost 7000 lines of lecture example code
  - Over 260,000 lines of student repository code
Big Picture Review
Course Topics

• Using the Shell
• The C language
  • Pointers & memory model
  • Heap allocation
  • Strings
• File I/O
• Header files and the compilation pipeline
• Makefile’s
• Process: fork, wait, waitpid and exec*
• Low level I/O: open, close, read, write, seek
• Pipes (pipe) and Redirection (dup2)
• Signals handlers and sending with kill()
• Sockets
  • Connecting clients
  • Bound and listening servers
  • Multiplexing I/O using select
What is systems programming?
A View of the System Stack

- Your Python Code
- Python Libraries
- CPython (compiler & VM)
- C Standard Library
- Operating System Kernel
- Device Drivers
- Hardware (CPU & Peripherals)
A View of the System Stack

Low-level C Programs

C Standard Library

Operating System Kernel

Device Drivers

Hardware (CPU & Peripherals)
A View of the System Stack

Kernel (CSC369)

Userspace (CSC209)
“The Unix philosophy”

• Write programs that do one thing and do it well
• Write programs to work together
• Write programs that handle text streams, because text is a universal interface
The shell is the connective tissue or glue for the Unix philosophy.
Memory Model
Address Space

64-bit Logical / Virtual Address

2^{64} - 1

Stack

Unused Logical Address Space

Dynamic Data

Static Data

Code

0
Input/Output leads to Computation
• Operating systems organize themselves around concepts such as:
  • Files & Directories
  • Processes
  • Signals
  • Sockets
  • etc.
Files (like memory) consist of a sequence of bytes
Processes are a means to co-exist peacefully with other programs
Signals allow programs to be interrupted and be informed about unexpected events
Sockets enable co-operating processes to communicate
The Assignments
A1 — wc209 and untar

• Using and writing small Unix utilities in and around the shell

• Reading and processing bytes from file-like inputs
  • Both text and binary files are just bytes

• Interpreting a structured file format like tar
Investigate behind the scenes how `malloc` and `free` actually do their work on your behalf.

Simultaneously managing data structures and memory.

Linked list in C using pointers.

Interpreting a design specification and seeing flaws.
A3 — Implementing a Shell

• Understanding the Unix system calls mechanisms for process creation (fork, exec*) and management (wait, waitpid)

• Using pipes (pipe) and input/output redirections (dup2)
A4 — Socket Servers

- Socket server setup

- Handling multiple concurrent connections (multiplexing I/O) with a `select` event loop

- *Architecture* and associated data structures of a non-trivial server application
Assorted Comments
Starter Code

(the only part you’re not responsible for is lexer.c and parser.c from A3)
Error Handling vs Error Recovery
System call return values
NULL terminated argument array

Midterm solutions, `exclp.c` and `execvp.c`
What’s next?
A lot of software is still written (and being written) in C
Systems programming comes in many shapes and sizes
Systems programming comes in many shapes and sizes

- Parallel programs using threads and other concurrency primitives

- Programs that can be running simultaneously on more than one CPU/processor/core!!
Systems programming comes in many shapes and sizes

- Operating system kernels
  - Directly interfacing with the hardware
  - Even more cold, harsh reality of the Real World
Systems programming comes in many shapes and sizes

- High performance network servers
- Handling massive amounts of traffic and a huge number of requests
C++
Other Kinds of C/C++ Usage

- Demanding 3D graphics game engines
- Power (battery) sensitive mobile apps
- Huge applications like web browsers and office suites
- Numerical simulation for scientific computing
Other Powerful Languages

- C++
- Python!
  - Look at the `socket` and `os` modules
- Go from Google (search for keyword `golang`)
- Rust from Mozilla
The End.

Thank you for a great summer!
Extra Diagram Slides
TCP Server

- socket()
- bind()
- listen()
- accept()

block until connection from client

- read()
- write()
- close()

TCP Client

- socket()
- connect()

Connection establishment (3-way handshake)

- write()

Data transfer

- read()
- close()

End-of-file notification
TCP Server

- `socket()`
- `bind()`
- `listen()`
- `accept()`

block until connection from client

TCP Client

- `socket()`
- `connect()`
- `write()`

Connection establishment (3-way handshake)

data transfer

TCP Server

- `read()`
- `write()`
- `close()`

data transfer

TCP Client

- `read()`
- `close()`

end-of-file notification
Compilation Process

* .c source code

C Compiler:

Pre-processor → Lexical Analysis → Parsing → Typecheck

Optimizations… → Codegen → Linking

Executable binary
calling process
pipefd[1] pipefd[0]

direction of data flow

Kerrisk figure 44-2
Kerrisk figure 44-3a: After `fork`
Kerrisk figure 44-3b: After closing unused descriptors
Kerrisk figure 44-3b: After closing unused descriptors