UNIVERSITY OF TORONTO SCARBOROUGH

Lecture 1: Course Overview
Introduction to Unix & Shell

Dr. Naureen Nizam
nnizam@cs.toronto.edu

CSCB09H3: Software Tools and Systems Programming (SY 110)
Department of Computer and Mathematical Science
Jan 7/9, 2018

UTSC Tomorrow is created here.
UNIVERSITY OF TORONTO SCARBOROUGH
1265 Military Trail, Toronto, Ontario M1C 1A4

Administrivia

- Email: nnizam@cs.toronto.edu
  - Subject must include **CSCB09:**
  - State your question clearly with enough context. Please include your full name and your utorid in the body of your email.
- Office Hours:
  - Wednesday 2 – 4 pm in IC building – IC400A
  - by email: any time
  - by appointment: email to arrange one
- **Tutorials starting next week!**

Course Website

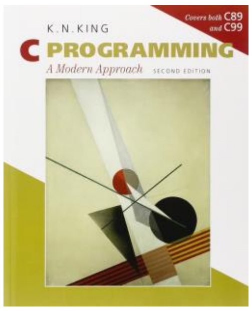
- Course website: Quercus
  https://q.utoronto.ca
  - Course Syllabus
  - Lecture slides
  - Announcements
  - Lab Exercises + Soln
  - Assignments
  - Discussion Board
  - Grades

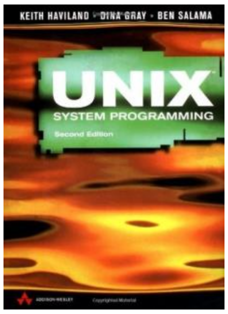
Course Overview

Software techniques in a Unix-style environment, using scripting languages and a machine-oriented programming language (typically C). What goes on in the system when programs are executed. Core topics: creating and using software tools, pipes and filters, file processing, shell programming, processes, system calls, signals, basic network programming.

- **Prerequisite:** CSCA48H3 and [CGPA 2.75 or enrolment in a CSC Subject POSt or enrolment in a non-CSC Subject POSt for which this course is needed to satisfy program requirements]
- **Exclusion:** CSC209H
- **Breadth Requirement:** Quantitative Reasoning

Course Readings





Course Format

The course will consist of the following:
- Lectures (10 min break mid-way)
- Labs
- Discussion Board
- Assignments X 3
- Midterm
- Final Exam

Evaluations

Components	Probable Topics	Weight	Due Date
Labs	Refer to the Course Outline	5%	Weekly
Assignment 1	Shell Use and Programming	10%	Jan 27, 2019 (11:59PM)
Midterm	Shell and C	20%	Lec 01: Feb 27, 2019 (in-class) Lec 30: Feb 25, 2019 (in-class)
Assignment 2	C and System Calls	10%	Mar 3, 2019 (11:59PM)
Assignment 3	Fork and Pipes	10%	Mar 31, 2019 (11:59PM)
Final Exam*	Everything	45%	April 10 - 27, 2019
Total		100%	

*** To pass the course you must receive at least 40% on the final exam**

Assignments

All code **must** work on the lab servers (BV473/mathlab) to receive full marks.

Code that does not compile will receive a grade of 0

Don't wait until the last day!

Labs/Tutorials

Starting next week, in BV 473

Work in pairs

Attendance

Solutions will be posted on Quercus at the end of the week

Submitting Assignments

You will be using SVN to manage and submit your assignments.

The repositories are already set up.
(Currently still empty …)

You should start learning how to use it as soon as possible. Start with the following:

```
svn co https://mathlab.utsc.utoronto.ca/svn/cscb09w19/{student utorid}
```

Do not wait until the last minute to try to commit your assignment for the first time.

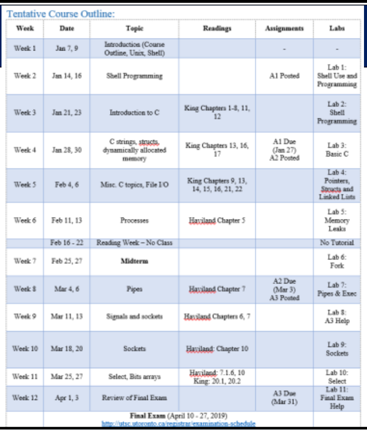
Submission of Work

All work must be submitted via SVN at 11:59 PM EST the day it's due.

All assignments must be submitted electronically via svn before 11:59 PM EST on the due date. Unless you have legitimate documentation for a late assignment, **10%** of the total value of the assignment will be deducted for **each day** that it is late. An assignment is considered late as soon as the due date and time passes (i.e., one minute past 11:59 PM EST), so you are strongly encouraged to submit your assignments early in order to avoid any technical delays. If you have legitimate documentation to explain your late assignment (e.g., a UTSC Medical Certificate completed by your doctor), you must contact Dr. Nizam as soon as possible to discuss your situation and to establish a new deadline for your assignment.

Course Overview

- Unix Software Tools
  - Understanding the shell
  - Shell programming
  - Make
- Systems Programming
  - C
  - files
  - processes
  - concurrency
  - communication


Tentative Course Outline:

Week	Date	Topic	Readings	Assignments	Labs
Week 1	Jan 7, 9	Introduction (Course Outline, Unix, Shell)		-	-
Week 2	Jan 14, 16	Shell Programming		A1 Posted	Lab 1: Shell Use and Programming
Week 3	Jan 21, 23	Introduction to C	King Chapters 1-8, 11, 12		Lab 2: Shell Programming
Week 4	Jan 28, 30	C strings, structs, dynamically allocated memory	King Chapters 13, 16, 17	A1 Due (Jan 27) A2 Posted	Lab 3: Basic C
Week 5	Feb 4, 6	Misc. C topics, File I/O	King Chapters 9, 13, 14, 15, 16, 21, 22		Lab 4: Pointers, Structs and Linked Lists
Week 6	Feb 11, 13	Processes	Haviland Chapter 5		Lab 5: Memory Leaks
	Feb 16 - 22	Reading Week – No Class			No Tutorial
Week 7	Feb 25, 27	**Midterm**			Lab 6: Fork
Week 8	Mar 4, 6	Pipes	Haviland Chapter 7	A2 Due (Mar 3) A3 Posted	Lab 7: Pipes & Exec
Week 9	Mar 11, 13	Signals and sockets	Haviland Chapters 6, 7		Lab 8: A3 Help
Week 10	Mar 18, 20	Sockets	Haviland: Chapter 10		Lab 9: Sockets
Week 11	Mar 25, 27	Select, Bits arrays	Haviland: 7.1.6, 10 King: 20.1, 20.2		Lab 10: Select
Week 12	Apr 1, 3	Review of Final Exam		A3 Due (Mar 31)	Lab 11: Final Exam Help
	Final Exam (April 10 - 27, 2019) http://utsc.utoronto.ca/registrar/examination-schedule				

Self Study Topics

- Using SVN
- Using basic Unix
- Learning an editor – vi, jove, emacs, scite, nedit, …
- Learning a debugger – ddd, gdb, eclipse
- Readings

Re-mark Policy

You may request a re-evaluation of term work if you believe it has been incorrectly or unfairly marked. You must make such requests within **one week** of receiving your marks back. The requests should be sent to the TAs in writing. Any section of an assignment in which the C program does not compile on the lab machines will receive a grade of 0. If you can explain clearly in a remarking request how to fix the problem (max of 10 lines of code), your program will be remarked with a **10%** penalty.

Please note, if a remark request is granted, the student must accept the resulting mark as the new mark, whether it goes up or down or remains the same.

Note*, the re-mark policy only applies to term work and excludes final examinations. Final Examination re-reads are handled directly by the Office of the Registrar.*

Academic Offenses (Plagiarism)

"The work you submit must be your own, done without participation by others. It is an academic offense to hand in anything written by someone else without acknowledgement."

You are not helping your friend when you *give* him or her a copy of your assignment. You are hurting your friend when you *ask* him or her to give you a copy of their assignment.

All suspected cases of academic dishonesty will be investigated following procedures outlined in the Code of Behaviour on Academic Matters. If you have questions or concerns about what constitutes appropriate academic behaviour or appropriate research and citation methods, you are expected to seek out additional information on academic integrity from your instructor or from other institutional resources (see http://www.utoronto.ca/academicintegrity/).

What is Cheating?

Cheating is
- copying parts or all of another student's assignment
- including code from books, web sites, other courses without attribution
- getting someone else to do substantial parts of your assignment
- giving someone else your solution

Cheating is not
- helping to find a bug in a friend's code (be careful)
- helping each other understand man pages or example code.

Access*Ability*

Students with diverse learning styles and needs are welcome in this course. In particular, if you have a disability/health consideration that may require accommodations, please feel free to approach me and/or the Access*Ability* Services Office as soon as possible. We will work with you and Access*Ability* Services to ensure you can achieve your learning goals in this course. Enquiries are confidential. The UTSC Access*Ability* Services staff (located in SW302) are available by appointment to assess specific needs, provide referrals and arrange appropriate accommodations (416) 287-7560 or ability@utsc.utoronto.ca.

Cell Phone and Laptop Usage

Please stay on task if you choose to use laptops or other mobile devices during class. These tools can be useful to take notes, refer to class readings, or look up important course concepts. However, checking social media, texting or other non-course specific activity distracts from your learning and can ultimately result in receiving a lower grade in this course.

Copyright

If a student wishes to tape-record, photograph, video-record or otherwise reproduce lecture presentations or slides, he or she must obtain the instructor's written consent beforehand. Without the consent, all such reproduction is an infringement of copyright and is absolutely prohibited. In the case of private use by students with disabilities, the instructor's consent will be granted.

Course materials, including lecture slides, are provided for the exclusive use of enrolled students. Do not share them with others.

Resources for you to succeed!

Academic Advising and Career Centre - http://www.utsc.utoronto.ca/aacc/

Centre of Teaching and Learning - http://www.utsc.utoronto.ca/ctl/

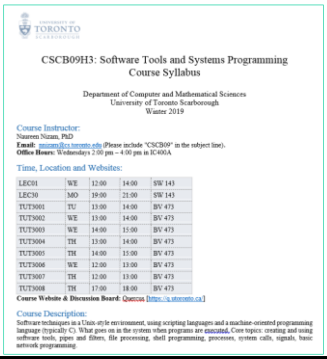
Library - https://utsc.library.utoronto.ca/

Registrar's Office - http://www.utsc.utoronto.ca/registrar/

The Writing Centre - http://www.utsc.utoronto.ca/twc/

Break (10 min)

Any Questions on Course Syllabus



UNIVERSITY OF TORONTO SCARBOROUGH

CSCB09H3: Software Tools and Systems Programming
Course Syllabus

Department of Computer and Mathematical Sciences
University of Toronto Scarborough
Winter 2019

Course Instructor:
Naureen Nizam, PhD
Email: nnizam@cs.toronto.edu (Please include "CSCB09" in the subject line).
Office Hours: Wednesdays 2:00 pm – 4:00 pm in IC400A

Time, Location and Websites:

LEC01	WE	12:00	14:00	SW 143
LEC30	MO	19:00	21:00	SW 143
TUT3001	TU	13:00	14:00	BV 473
TUT3002	WE	13:00	14:00	BV 473
TUT3003	WE	14:00	15:00	BV 473
TUT3004	TH	13:00	14:00	BV 473
TUT3005	TH	14:00	15:00	BV 473
TUT3006	WE	12:00	13:00	BV 473
TUT3007	TH	12:00	13:00	BV 473
TUT3008	TH	17:00	18:00	BV 473

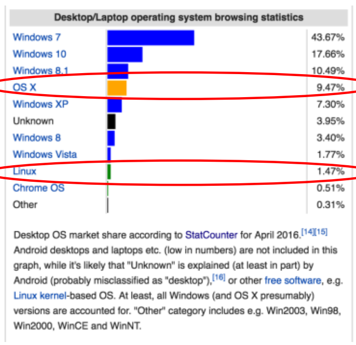
Course Website & Discussion Board: Quercus [https://q.utoronto.ca/]

Course Description:
Software techniques in a Unix-style environment, using scripting languages and a machine-oriented programming language (typically C). What goes on in the system when programs are executed. Core topics: creating and using software tools, pipes and filters, file processing, shell programming, processes, system calls, signals, basic network programming.

The plan for today..

- What is Unix and why are we using it?
- What is an operating system?
- What are files and directories?
- What is a shell and why do I care about shell programming?
- Along the way: learn/refresh basic Unix commands

Is anybody using Unix/Linux?



Desktop/Laptop operating system browsing statistics	
Windows 7	43.67%
Windows 10	17.66%
Windows 8.1	10.49%
OS X	9.47%
Windows XP	7.30%
Unknown	3.95%
Windows 8	3.40%
Windows Vista	1.77%
Linux	1.47%
Chrome OS	0.51%
Other	0.31%

Desktop OS market share according to StatCounter for April 2016.[14][15] Android desktops and laptops etc. (low in numbers) are not included in this graph, while it's likely that "Unknown" is explained (at least in part) by Android (probably misclassified as "desktop"),[16] or other free software, e.g. Linux kernel-based OS. At least, all Windows (and OS X presumably) versions are accounted for. "Other" category includes e.g. Win2003, Win98, Win2000, WinCE and WinNT.

Is anybody using Unix/Linux?

- %age of laptops/desktops running Unix?
  - 10%
- %age of tablets/smartphones running Unix?
  - 85%
- %age of web servers running Unix?
  - 65%
- %age of supercomputers running Unix?
  - 98%

facebook Google IBM twitter amazon

Why Unix?

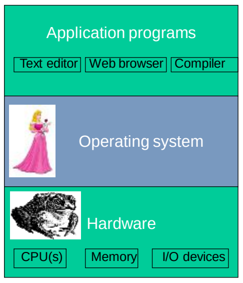
- 1969:
  - Ken Thompson at Bell Labs wanted to play Space Travel on his DEC PDP-7
  - Wrote the first version in assembler in one month
- Now:
  - Available on many platforms
  - Multi-user, multi-programmed
  - Good at
    - sharing computer resources
    - manipulation of files, processes, programs
    - inter-process and inter-machine communication

Why Unix?

LIVE FREE OR DIE
http://www.UNIX-systems.org
UNIX®
UNIX is a Registered Trademark of The Open Group

- Open source
  - Free
  - Access to operating features
  - Can be customized

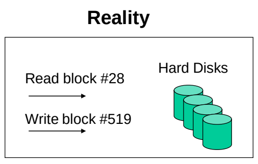
What is an operating system?

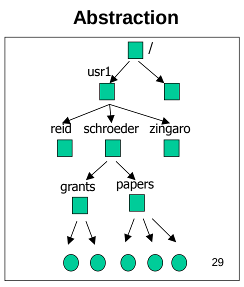


<u>The software layer between user applications and hardware.</u>

- Serves as a resource manager
  - allows proper use of resources (hardware, software, data)
- Serves as a control program (protection)
  - controls execution of user programs to prevent errors and improper use of the computer
- Turns ugly hardware into beautiful abstractions (provides services)

One such abstraction: Files and Directories





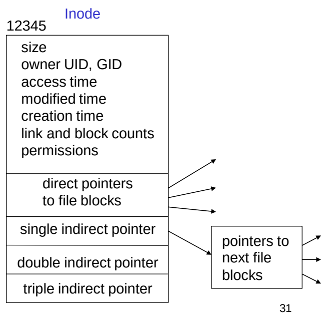
- Unix commands to navigate directories and files:
  - `ls, cd, pwd, cat, more`
- (Use man pages to learn more about commands, e.g. type "`man ls`" for info on the ls command)

What is a file?

- A named collection of data with some attributes
  - Name
  - Owner (user and group)
  - Size
  - Permissions
  - Time of creation, last access, last modification
  - Location on disk
- How to get information on a file in Unix?
  - `ls -l, stat`
- Some other Unix file operations?
  - Remove (`rm`), move (`mv`), copy (`cp`), count words/lines in file (`wc`), list its contents (`cat, more`), change permissions (`chmod`)

How does Unix implement a file?

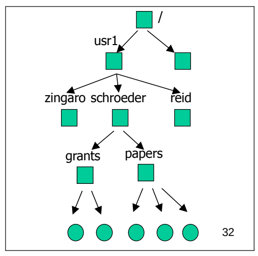
- A data structure called **inode** stores the information about a file, including which disk blocks contain the file data.
- Identified by its **inode number**



What is a directory?

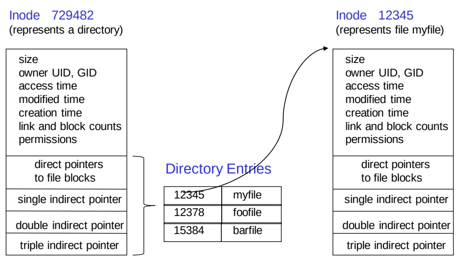
- A collection of files (and sub-directories)
- One special directory: the root directory (named “/”)
- In Unix: every directory is a file!

Abstraction



What is a directory?

- Internally, every directory is a file itself!
  - File with special content: *directory entries*
  - A directory entry maps a file name to an inode.



12345	myfile
12378	foofile
15384	barfile

Many more things are files …

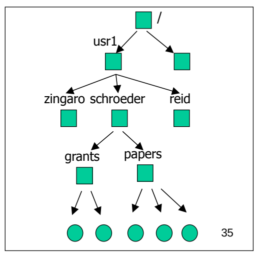
- “Everything is a file.”
- Unix provides a file interface for all Input/Output.
  - regular files
  - directories
  - devices
    - video (block)
    - keyboard (character)
    - sound (audio)
    - network (block)
- File interface = open, read, write, close

Try `ls -l /dev` and look at the permissions string.
`crw-------`
`brw-------`
c = character, b = block

The directory hierarchy

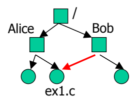
- The directory tree is not really a tree as shown in the previous pictures…
  - But an acyclic graph
  - Why?

Abstraction



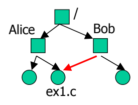
The directory hierarchy

- Sharing of files can be implemented by creating a new directory entry called a ***link*** : a **pointer** to another file or subdirectory
- An “`ls`” in Bob’s home directory will show ex1.c



Hard Links

- Create with “`ln <target> <name of link>`”
- Second directory entry identical to the first
- What happens if anybody removes (`rm`) ex1.c?
  - A file is only removed if it no longer has any name/ hard links.



‘~Alice’ directory

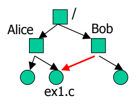
File Name	Inode	Type
…	…	…
ex1.c	42	file
…	…	…

‘~Bob’ directory (hard link)

File Name	Inode	Type
…	…	…
ex1.c	42	file
…	…	…

Soft Links

- Create with “`ln -s <target> <name of link>`”
- Directory entry points to small file containing the true path of the linked file
- What happens if anybody removes (`rm`) ex1.c?
  - If Alice removes ex1.c, dangling reference
  - If Bob removes ex1.c, only the entry in his directory is removed



‘~Alice’ directory

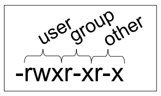
File Name	Inode	Type
…	…	…
ex1.c	42	file
…	…	…

‘~Bob’ directory (hard link)

File Name	Inode	Type
…	…	…
ex1.c	529	link
…	…	…

“/Alice/ex1.c”

File Permissions

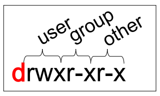


Ch 3.1

```
-rwxr-xr-x 1 nizamnau  cmsusers   0 Jan   6 11:35 file1
-r--r--r-- 1 nizamnau  cmsusers   0 Jan   6 11:35 file2
-rw------- 1 nizamnau  cmsusers   0 Jan   6 11:35 file3
```

- Each file is owned by a particular user and group
- Each file has three permission entries
  - Permissions for owning user
  - Permissions for owning group
  - Permissions for all other users
- Each entries specify three permissions:
  - read, write, execute – pretty much what you think

Directory Permissions



Ch 3.1

```
drwxr-xr-x 2 nizamnau  cmsusers 512 Jan  6 11:36 dir1/
dr-x--x--x 2 nizamnau  cmsusers 512 Jan  6 11:36 dir2/
dr--r--r-- 2 nizamnau  cmsusers 512 Jan  6 11:36 dir3/
```

- Directory permissions
  - read – you can run `ls` on the directory
  - write – you can create and delete files in the directory
  - execute – you can “pass through” the directory when searching subdirectories (you can `cd` into it).

Ch 3.1

Directory Permissions

- r without x:
  - can run ls to see filenames but not access files or cd into the directory
- x without r:
  - access file if you know it exists (can cd into directory, but not run ls)
- You can delete a file in a directory with w permissions, even without permissions on file itself!

How can you change permissions?

- `chmod`
- Lots of ways, check `man chmod`
- E.g. give user owning file `fname` execute permissions:
  `chmod u+x fname`
- E.g. give all users in group read permissions
  `chmod g+r fname`
- E.g. give all users all permissions
  `chmod a+rwx fname`

What is a shell?

- Commandline interpreter
- Interface between user and OS
- The shell is a program that:
  - Waits for input commands
  - Analyzes command
  - Determines what actions are to be performed
  - Performs the actions

Which shell?

- sh – Bourne shell
  - Most common, other shells are superset
  - Good for programming
- csh or tcsh
  - C-like syntax
  - Some argue not suitable for programming (“csh considered harmful”)
- bash – Bourne again shell
  - Based on sh with some csh features
- korn – commercial product

Which shell?

- On mathlab and the linux lab bash is default
- Bash is a superset of sh
- For B09, we will only cover features that belong to sh.

- All OS’s have graphical user interfaces, why would you use the shell?

Shells are very powerful …

What can shells do for you?

- Execute all the commands we have already seen (ls, cd, pwd, rm, mv, man, chmod, wc,..)
- A few other very useful features:
  - I/O redirection
  - Pipelining of commands
  - Filtering output of commands
  - Job control
- Automate things (shell programming)

Input and output redirection

- By default, programs read from standard input (keyboard), write results on standard output (screen), and write errors to standard error (screen)
- You can redirect input, output and errors:

  “ > filename” redirects output to file

  “>> filename” appends output to file

  “< inputfile” redirects input (reading from inputfile instead of keyboard)
- 1 means stdout, 2 means stderr

- `ls >output.txt`   # saves the output of ls in output.txt
- `ls -z >output.txt`   # does not save output in output.txt …
- `ls -z 2>output.txt` # saves output in output.txt

Pipelines

- Use "|" to send the output of one command to the input of another command
- E.g. to count the number of lines produced by "`ls -l`":

```
ls -l | wc -l
```

- E.g. to count the number of processes associated with current terminal:

```
ps | wc -l
```

(ps is a command that lists running processes)

Filters

- A filter reads from standard input, processes the input and writes on standard output
- Some useful filters (read about them using `man`)
  - `wc:` count words, lines, characters
  - `grep:` filter lines that do or do not match a pattern
  - `uniq:` remove repeated lines
  - `sort:` sorts input
  - `head:` output only the first lines of the provided input
  - `tail:` output only the last lines of the provided input
  - `sed:` a stream editor to perform text transformations

Job control

- A job (or process) is program in execution
  - Use `ps` to view processes
- Foreground job: has control of the terminal
- Background job: runs concurrently with shell in the background
  - To run a program in the background append & to the name of the program
- At any point a program can be running or suspended
  - Hit `<ctrl> z` to suspend the current foreground job

Job control

- `jobs` gives you a list of jobs; each is associated with a job number
- `fg [num]` puts job `num` in the foreground
- `bg [num]` puts job `num` in the background
- `kill %num` kills job `num`

That's it for today!