EECS2031 A,C Software Tools

F 2021

Sep 8,9 2021 Lecture 1.

1

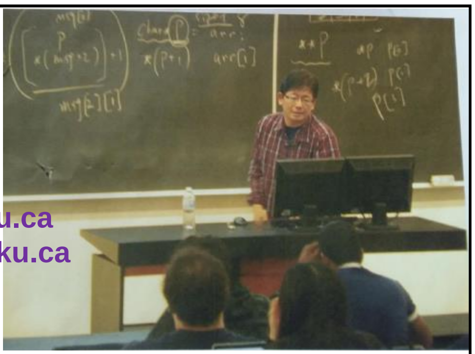
The Course EECS2031 Software Tools

- Lectures:
  - Week: Wed ~ Tuesday, 12 weeks.
  - Wednesdays 8:30 - 9:30 (A), Thursdays 19:00-20:00 (C)
  - Fridays: 8:30 - 9:30 (A), Mondays 19:00-20:00 (C)
  - Zoom live meeting
  - Recorded. Video on Echo 360 (link on eClass)
  - Keep muted, Raise hand, or chat with questions.
    - 'Real-time' question asap, other later
  - Interrupt me for critical glitches.
    - Loss of audio, no screen sharing
- Labs: Wednesdays, Thursdays evening (more later)
- Course website: eClass (formerly Moodle)
  - Zoom, video link, slides, labs, forums, announcements, slu ….
  - Visit frequently

3

3

The instructor



- Dr. Hui Wang
  - Office: LAS (CSEB) 2015
  - Email: **huiwang@cse.yorku.ca**
    huiwang@eecs.yorku.ca
- Office hours (tentative)
  - Mondays 10:00 ~ 10:30 in LAB link
  - Tuesdays 10:00 ~ 10:30 in LAB link
  - Wednesdays, 20:30 ~ 21:00 in LAB session
  - Thursdays, 21:30 ~ 22:00 in LAB session
  - After class on Zoom
  - By appointments

OFFICE HOURS

- COSC/CSE/EECS2031 student, TA, instructor

YORK UNIVERSITÉ UNIVERSITY

4

4

Course content

- Introduces software tools that are useful in the software development process.
- You will be exposed to the layers between a programming language and the operating system and the CPU.

The course covers the following topics:

- ANSI-C
  - Learning how to write C programs
  - *C Basics, stdio, pointers, memory management, C libraries*
- Unix (Linux) operating system
  - Using Unix tools to automate compilation, execution and testing
  - *Commands/utilities, filters and pipes, Shell programming under Unix -- Bourne (again) shell scripts*
- Testing and debugging

YORK UNIVERSITÉ UNIVERSITY

5

5

Why 'Software Tools', why now

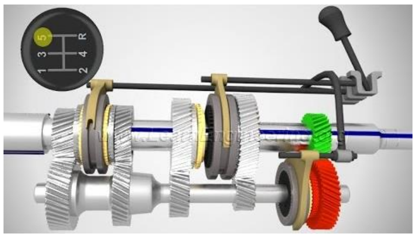
- EECS 1011/1012 → 1021/1022 → 2030
  - Basic programming skills / concepts
  - read API specification (client)
  - implement API (implementer)
- Java gives you a "safe" programming environment (VM)
  - higher level

- Now good time to learn how to deal with lower layers (memory, CPU management etc).
  - Some domains require working on lower layer
  - Better understanding of higher layer
  - Lay foundation for future courses, researches, careers, ……

6

6

Why C and Unix



Right platforms to teach skills necessary for practical program development

- C: Expose students to underlying layers/raw machine
  - C's ability to handle low-level activities (direct memory access, memory allocation etc)
  - Safety layers not present (C has poor error detection and significantly fewer safeguards than Java)
    - A good language to learn testing and debugging
- Unix: where C naturally runs.
  - Good environment to learn systematic testing

YORK U UNIVERSITÉ UNIVERSITY

7

7

Course learning outcome (CLO)

- Use the basic functionality of the Unix shell, such as standard commands and utilities, input/output redirection and pipes
- Develop and test Unix shell scripts of significant size
- Develop and test programs written in the C programming language
- Describe the memory management model in the C programming language
- Use test, debug and profiling tools to check the correctness of programs

8

8

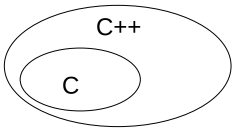
Course objective

- By the end of the course, you should be able to
  - Develop modest-sized programs in C
  - Use UNIX shell commands/utilities
  - Develop programs using UNIX shell scripting language
  - Test and debug C and other programs using UNIX command/scripts

9

What you can actually gain

- C, Unix for courses/researches/careers/
  - Computer Organization, OS, Embedded system courses
  - Work/research on Networking, Embedded systems, image processing…
  - ....
- Better understanding of programming, including Java
  - `z = a++;     z += 2;     c= a>b ? d: e;     c >>= 2;`
  - `Student s1 = s2;   s1.age++;   s2.age ?`
  - `"Shallow copy" vs. "Deep copy"`
  - `"Pass/call by value" vs. "Pass/call by reference"`
- Automatically learn some C++ !!!
  - Lots opportunities



10

10

Some Applications

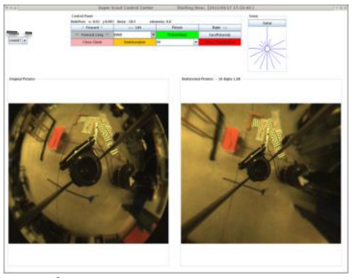
- Embedded systems, Network, image processing …..
- An example -- Driving robots
  - Robot side: Unix (Ubuntu), C, shell script, make file ….
  - Base station side: Java, Python, ….
  - Communication: Socket programming

11

11

Wireless network
socket

C
Unix/Ubuntu



Java, Python
Unix/Window/Mac

12

12

rovers

C++, Python
on
Unix/Ubuntu
(ROS)

13

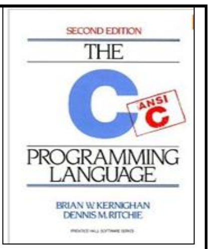
13

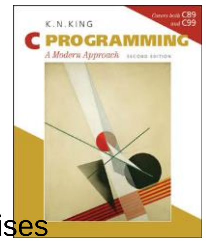
drones

C++, Python
on
Unix/Ubuntu
(ROS)

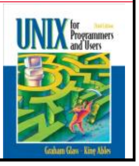
14

Textbooks (recommended)





- ***The C Programming Language (2nd Ed)***
  - B. W. Kernighan and D. M. Ritchie (K&R)
  - ISBN 0-13-110362-8
  - Short and well-written, covering ANSI-C (C89)
  - Classic C book. Bible
- ***C programming: A modern approach*** *(2nd)*
  - Well written. 800 pages > $150
- ***zyBooks: programming in C with zyLabs***
  - Online book with interactive contents and exercises
  - email support@zybooks.com ~$50 USD for 6 month?



- Unix: another 800 pages book …
- List of other recommended books on course web

15

Administrivia – Assessments

- Components and weights (Tentative, so far)
  - ≈ 13% – Weekly labs (6~7)
    - 1~3% each
  - ≈ 28% – Term test
    - Online, written (18%) and coding (10%)
    - End Oct, mid Nov on two Friday nights
  - ≈ 5% – Subject matter quizzes (“Participation Activity”)
    - On Tuesday nights
  - ≈ 16% – two programming assignments
  - ≈ 38% – Final exam
    - Online

Disclaimer: Subject to adjustment in the near future.

YORK UNIVERSITÉ UNIVERSITY

17

17

Administrivia

- Weekly Labs/exercises
  - Released after (some) lectures.
  - Each lab weights 1~3%.
  - Due in about a week
  - Hands-on learning process for the preceding lecture.
    - Open book. Discussion allowed. Ask questions to TAs or me!
    - Prof. or TA will be on duty, one-to-one, optional
  - Wednesdays 19:00 ~ 21:00
  - Thursdays 20:00 ~ 22:00
  - I will be there (also my office hour)
    - Last half hour in LAB session
  - Check schedule on ‘Weekly Labs’ page.

Mondays 10:00 ~ 10:30
Tuesdays 10:00 ~ 10:30
By myself

YORK UNIVERSITÉ UNIVERSITY

19

19

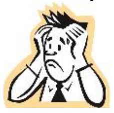
Administrivia

- Subject Matter Quizzes (discussion not allowed)
  - Small exercise for previous lecture
  - get feedback; ~~motivation for attendance~~; practice for test/exam
- Term test
  - Online, writing and coding, (discussion not allowed)
  - Writing: Oct 29 (Nov 5?) evening 7~10.
  - Coding: Nov19 evening 7~10
- Assignments
  - Larger programing “exam”, individual (discussion not allowed)
  - About 10~14 days
  - More details later …
- Final exam
  - Online (discussion not allowed)
  - More details later …
  - Date unknown yet

20

20

Overall, Challenging but doable course



- C/Java bit shifting, (notoriously scary) pointers and memory allocation
  - `int i = k >> 5;`
  - `int * p = &x;    int * pArr[3];`
  - `current -> next = (int *) malloc(sizeof(struct node))`
- Unix utilities: `find . -name *.c  -exec chmod 762 {} \;`
- Unix shell syntax bit strange and strict

```
count=1
while [ $count -lt 100 ]
do
    count=`expr $count + 1`
    echo $count
done
```

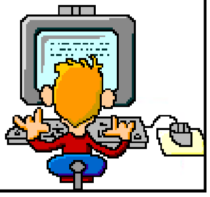
21

21

Useful suggestions

- Come to the lectures
- Watch videos, read the lecture notes and the textbook!
  - Following notes. Use recommended textbooks for details
  - Videos, Notes will be finalized shortly after class

- Do the labs and assignments on your own!
  - Discussion allowed only for labs. Not for others
  - (allowed) discussion != collaboration != sharing solutions
- Don't be shy to ask for help
  - come to the lab session, office hour
  - eClass forum
  - email me (specify "2031" , EECS username)



- Practice, practice, and practice!

22

22

Academic Integrity

- Honesty, originality and academic integrity matters to us.
- Plagiarism and cheating are not tolerated!
- Read https://lassonde.yorku.ca/academic-integrity for the consequences. Read the slides on eClass
- Weekly labs: discussion ✓
  - Discussion != sharing solutions
- Assignments, tests, exam:
  - Discussion not allowed ✗

YORK U UNIVERSITÉ UNIVERSITY

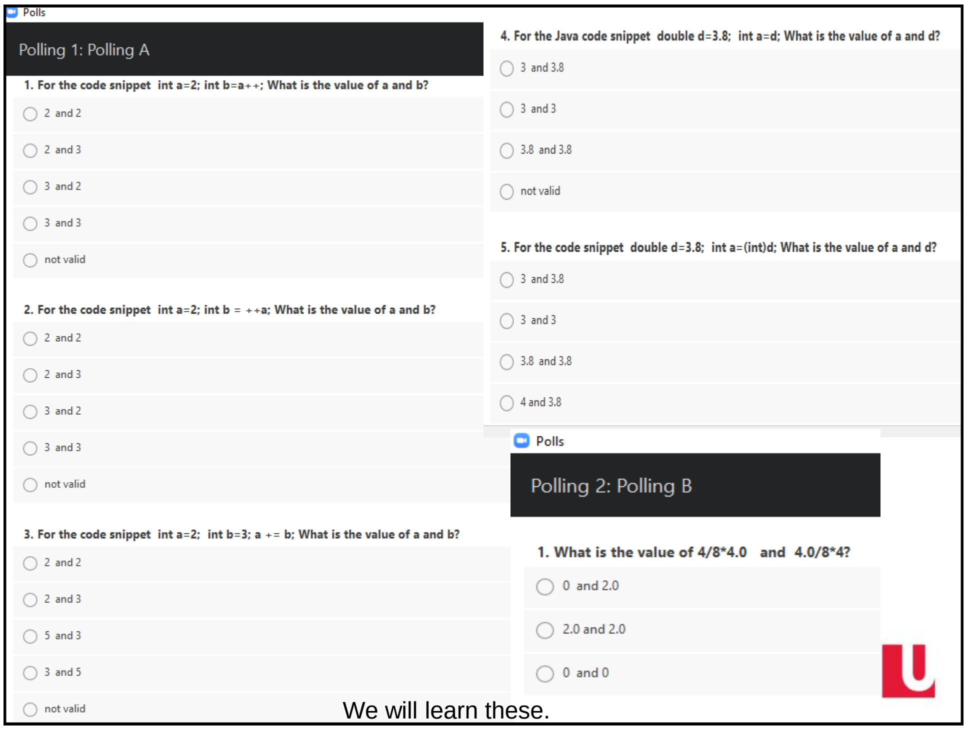
23

23

- Any questions so far

24

24



Polls

Polling 1: Polling A

1. For the code snippet int a=2; int b=a++; What is the value of a and b?
   - ○ 2 and 2
   - ○ 2 and 3
   - ○ 3 and 2
   - ○ 3 and 3
   - ○ not valid

2. For the code snippet int a=2; int b = ++a; What is the value of a and b?
   - ○ 2 and 2
   - ○ 2 and 3
   - ○ 3 and 2
   - ○ 3 and 3
   - ○ not valid

3. For the code snippet int a=2; int b=3; a += b; What is the value of a and b?
   - ○ 2 and 2
   - ○ 2 and 3
   - ○ 5 and 3
   - ○ 3 and 5
   - ○ not valid

4. For the Java code snippet double d=3.8; int a=d; What is the value of a and d?
   - ○ 3 and 3.8
   - ○ 3 and 3
   - ○ 3.8 and 3.8
   - ○ not valid

5. For the code snippet double d=3.8; int a=(int)d; What is the value of a and d?
   - ○ 3 and 3.8
   - ○ 3 and 3
   - ○ 3.8 and 3.8
   - ○ 4 and 3.8

Polls

Polling 2: Polling B

1. What is the value of 4/8*4.0 and 4.0/8*4?
   - ○ 0 and 2.0
   - ○ 2.0 and 2.0
   - ○ 0 and 0

We will learn these.

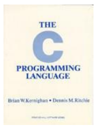
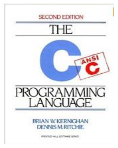
26

Overview of C K&R ch1.1-1.8, ch7.1-7.4

- System programming language
  - Originally used to write Unix and Unix tools
  - Later also a popular application programming language
- History of C

BCPL → B → C → K&R C → ANSI C (C89/90) → C99 → C11
1960 1970 1972 1978 1988-89 1999 2011

(NB)

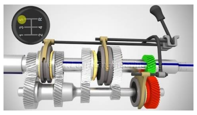



- ANSI-C (C89) standard by *American National Standard Institute*

27

27

Overview of C K&R ch1.1-1.8, ch7.1-7.4



- Features
  - Low level
    - with high-level constructs and ability to handle low-level activities (direct memory access, memory allocation etc.)
  - Small
    - limited set of features and library functions. LinkedList?
  - Procedural -- Data completely separate from Methods

- Strengths
  - Efficient and <u>fast</u>
  - Integration with UNIX
- Weaknesses
  - Permissive, Error-prone `int i=3.2;` `if (x=1) …`

Not OK in Java
But OK in C

28

28

Overview of C

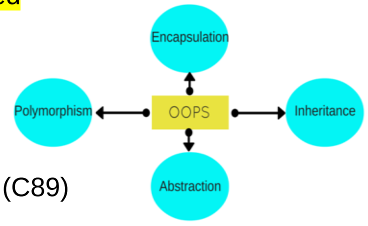
- Predecessor of modern object oriented languages
  - Many languages derived from C (e.g., C++, Java, Objective-C)
  - $$\begin{tikzcd} C \arrow[r] & C{+}{+} \arrow[r] \arrow[rr, bend right] & Java \arrow[r] \arrow[r, bend right] & C\# \end{tikzcd}$$
- Syntax of C
  - Something same as or similar to Java (adopted in C++/Java)
    - Variable, data type, operator (arithmetic, relational, logical etc), operation precedence, expressions, flow control, …
      - `int, double, int i = 2; i++;  i += 2;`
      - `if else, for…, while, do while, switch,`
  - Something different from Java (not adopted)

29

29

Overview of C

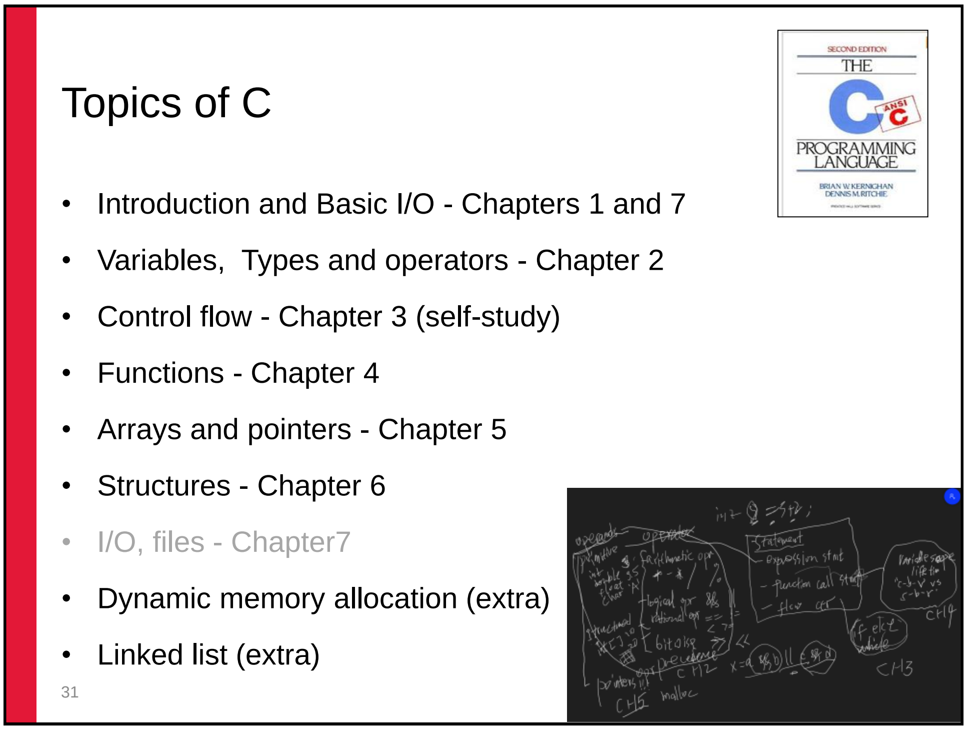
- Some syntactic differences against Java
  - Procedure-oriented vs. Object Oriented
    - ✓ No classes, objects
    - ✓ No E, I, P (?)
  - No garbage collection
  - No Exceptions (try - catch)
  - No //, only /* */ multi line – for ANSI C (C89)
  - No type **`String`**
  - No type **`boolean`** – for ANSI C (C89)
  - Declare or define a function before its first use
  - Declare all variables at the block beginning -- for ANSI C (C89)
  - Has (explicit) pointers
  - Can do (low level) memory allocation and de-allocation
  - Pre-processing, header files, global variables
  - ……



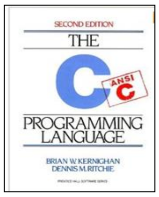
Additional resources on website

30

30

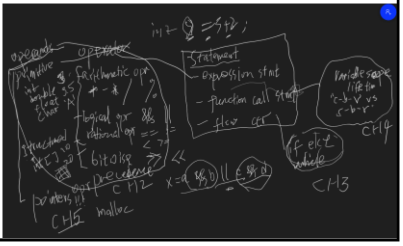


Topics of C



- Introduction and Basic I/O - Chapters 1 and 7
- Variables, Types and operators - Chapter 2
- Control flow - Chapter 3 (self-study)
- Functions - Chapter 4
- Arrays and pointers - Chapter 5
- Structures - Chapter 6
- I/O, files - Chapter7
- Dynamic memory allocation (extra)
- Linked list (extra)

31



31