


## TCSS 422: OPERATING SYSTEMS


### INTRODUCTION

**Wes J. Lloyd**  
 School of Engineering and Technology  
 University of Washington - Tacoma



September 26, 2018    TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    Tacoma

## OBJECTIVES



- Syllabus, Course Introduction
- C Review
- Demographics Survey
- Chapter 4: Operating Systems – Three Easy Pieces
  - Introduce operating systems
  - Management of resources
  - Concepts of virtualization/abstraction
  - CPU, Memory, I/O
  - Operating system design goals

September 26, 2018    TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.2

## TCSS422 – FALL 2018 COMPUTER OPERATING SYSTEMS

- Syllabus
- Grading
- Schedule
- Assignments

September 26, 2018    TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.3

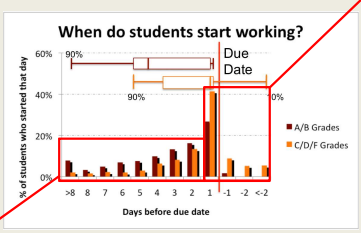
## TCS422 COURSE WORK

- **Assignments**
  - Assignment 0: Linux /scripting
  - Assignments 1 – 3 (4): roughly every two weeks
- **Quizzes**
  - ~ 6-8 quizzes
  - Drop lowest two
  - Variety of formats: in class, online, reading, tutorial / activity
- **Exams: Midterm and Final**
  - Two pages of notes, calculator
  - Final exam is comprehensive, with emphasis on new material

September 26, 2018    TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.4

## TCSS 422: PROGRAM DUE DATES

■ **Programs - please start early:**



From Virginia Tech Department of Computer Science - 2011

September 26, 2018    TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.5

## TCSS 422: PROGRAM DUE DATES

■ **Programs - please start early**

- Work as if deadline is several days earlier
- Allows for a “buffer” for running into unexpected problems
  - Underestimation of the task at hand
  - Allows time to seek C help from SCI 106/108 lab mentors
- If less familiar with C/pointers (TCSS 333), **BUDGET MORE TIME**

September 26, 2018    TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.6

## UBUNTU 18.04 – VIRTUAL MACHINE

- Ubuntu 18.04
  - Open source version of Debian-package based Linux
  - Package management: “apt get” repositories
    - See: <https://packages.ubuntu.com/>
- Ubuntu Advantages
  - Enterprise Linux Distribution
  - Free, widely used by developers
  - Long term releases (LTS) every 2 years, good for servers
  - 6 month feature releases, good for sharing new features with the community

September 26, 2018
TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.7

## UBUNTU 18.04 – VIRTUAL MACHINE INSTALLATION

- Ubuntu 18.04 on Oracle VirtualBox
- HOW-TO installation videos:
  - Windows 10
    - <https://www.youtube.com/watch?v=QbmRXJKsvs>
  - Mac OS X (not specific to 18.04)
    - <https://www.youtube.com/watch?v=sNixOS6mHIU>
- Guest Additions
  - Provides file system sharing, clipboard integration, mouse tricks
- <https://www.youtube.com/watch?v=qNecdUstuTPw>

September 26, 2018
TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.8

## C PROGRAMING IN TCSS 422

- Many OSES are coded primarily in C and Assembly Language
- Computerworld, 2017 Tech Forecast Survey

What legacy platforms do you still support and hire for?

None	65%
DB2	13%
C	10%
Cobol	9%
Assembly language	8%
Perl	5%
Delphi Object Pascal	3%
Fortran	3%
REXX	3%
Pascal	2%
Other	9%

September 26, 2018
TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.9

## C MENTORING

- <https://www.tacoma.uw.edu/institute-technology/student-support-workshops-mentors>
- School of Engineering and Technology Mentors
- Located in Science 106 / 108 Labs
- Monday – Thursday: ~9:30 am – 7:30 pm
- Friday: ~ 11-3pm
- Fall quarter hours to be posted

September 26, 2018
TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.10

## INSTRUCTOR HELP

- Office hours: to be announced - set by survey results
  - Also available by appointment
- End of class: good for quick questions, assignment Q&A
- It will be difficult to tutor all students individually on C
- Take **ownership** of your educational outcome
  - 11 weeks spent in TCSS 422 is very small relative to entire IT career
  - Make the most of this **limited** opportunity
    - Maximize your educational investment
  - **\*\*\* Ask questions in class \*\*\***
  - Also questions after class, email, Canvas discussion boards
  - Seek help using UWT resources, the internet, YouTube videos (video.google.com) and online tutorials

September 26, 2018
TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.11

## CLASS PARTICIPATION

- **Questions and discussion are strongly encouraged**
  - Leverage your educational investment
  - All questions are encouraged! All are good!
  - This instructor does not mind repeat questions – better to be sure than sorry!
- **Daily feedback surveys**
  - How much is new vs. review?
  - Checking the pace...
  - What is unclear? It's helpful to know when topics are not clear
  - Use the survey to write questions and feedback that come to you during the lecture
- **Pol-EV**

September 26, 2018
TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.12

# C REVIEW



September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma L1.13


# DEMOGRAPHICS SURVEY



<http://faculty.washington.edu/wlloyd/courses/tcss422/announcements.html>

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma L1.14

# INTRODUCTION TO OPERATING SYSTEMS



September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma L1.15

# VIRTUAL MACHINE SURVEY

- Please complete the Virtual Machine Survey to request a "School of Engineering and Technology" remote hosted Ubuntu VM
- <https://goo.gl/forms/zjmEk0FV9ctDFIDI2>

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma L2.16

# OBJECTIVES

- **Chapter 2: Operating Systems – Three Easy Pieces**
  - Introduction to operating systems
  - Management of resources
  - Concepts of virtualization/abstraction
  - **THREE EASY PIECES:**
    - Virtualizing the CPU
    - Virtualizing Memory
    - Virtualizing I/O
  - Operating system design goals

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma L1.17

# OPERATING SYSTEMS

- Responsible for:
  - Making it easy to **run** programs
  - Allowing programs to **share** memory
  - Enabling programs to **Interact** with devices

**OS is in charge of making sure the system operates correctly and efficiently.**

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma L1.18

## RESOURCE MANAGEMENT

- The OS is a resource manager
- Manages CPU, disk, network I/O
- Enables many programs to
  - **Share** the CPU
  - **Share** the underlying physical memory (RAM)
  - **Share** physical devices
    - Disks
    - Network Devices
    - ...

September 26, 2018
TCCS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.19

## VIRTUALIZATION

- Operating systems present **physical resources** as **virtual representations** to the programs sharing them
  - Physical resources: CPU, disk, memory, ...
  - The virtual form is "**abstract**"
  - The OS presents an illusion that each user program runs in isolation on its own hardware
  - This virtual form is general, powerful, and easy-to-use

September 26, 2018
TCCS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.20

## ABSTRACTIONS

- What form of abstraction does the OS provide?
  - **CPU**
    - Process and/or thread
  - **Memory**
    - Address space
    - → large array of bytes
    - All programs see the same "size" of RAM
  - **Disk**
    - Files

September 26, 2018
TCCS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.21

## WHY ABSTRACTION?

- Allow applications to reuse common facilities
- Make different devices look the same
  - Easier to write common code to use devices
    - Linux/Unix Block Devices
- Provide higher level abstractions
- More useful functionality

September 26, 2018
TCCS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.22

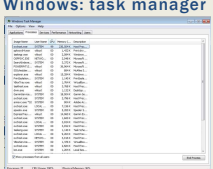
## ABSTRACTION CHALLENGES

- What level of abstraction?
  - How much of the underlying hardware should be exposed?
    - What if **too much**?
    - What if **too little**?
- What are the correct abstractions?
  - Security concerns

September 26, 2018
TCCS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.23

## VIRTUALIZING THE CPU

- Each running program gets its own "virtual" representation of the CPU
- Many programs seem to run at once
- Linux: "top" command shows process list
- Windows: task manager



```

top - 09:27:47 up 100 days, 1:11, load average: 0.00, 0.00, 0.00
  PID TID          PR  NI  TTY          STAT       S   CPU   MEM
  1    1    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  2    2    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  3    3    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  4    4    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  5    5    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  6    6    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  7    7    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  8    8    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
  9    9    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 10   10    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 11   11    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 12   12    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 13   13    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 14   14    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 15   15    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 16   16    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 17   17    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 18   18    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 19   19    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 20   20    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 21   21    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 22   22    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 23   23    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 24   24    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 25   25    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 26   26    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 27   27    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 28   28    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 29   29    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 30   30    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 31   31    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 32   32    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 33   33    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 34   34    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 35   35    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 36   36    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 37   37    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 38   38    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 39   39    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 40   40    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 41   41    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 42   42    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 43   43    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 44   44    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 45   45    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 46   46    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 47   47    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 48   48    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 49   49    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 50   50    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 51   51    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 52   52    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 53   53    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 54   54    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 55   55    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 56   56    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 57   57    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 58   58    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 59   59    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 60   60    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 61   61    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 62   62    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 63   63    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 64   64    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 65   65    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 66   66    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 67   67    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 68   68    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 69   69    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 70   70    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 71   71    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 72   72    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 73   73    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 74   74    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 75   75    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 76   76    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 77   77    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 78   78    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 79   79    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 80   80    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 81   81    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 82   82    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 83   83    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 84   84    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 85   85    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 86   86    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 87   87    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 88   88    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 89   89    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 90   90    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 91   91    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 92   92    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 93   93    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 94   94    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 95   95    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 96   96    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 97   97    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 98   98    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
 99   99    00000000    0   0   pts/0        Ss         0   0.0%   0.0%
100  100    00000000    0   0   pts/0        Ss         0   0.0%   0.0%

```

September 26, 2018
TCCS422: Operating Systems [Fall 2018]  
School of Engineering and Technology, University of Washington - Tacoma
L1.24

## VIRTUALIZING THE CPU - 2

■ Simple Looping C Program

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/time.h>
4  #include <assert.h>
5  #include "common.h"
6
7  int
8  main(int argc, char *argv[])
9  {
10     if (argc != 2) {
11         fprintf(stderr, "usage: cpu <string>\n");
12         exit(1);
13     }
14     char *str = argv[1];
15     while (1) {
16         Spin(1); // Repeatedly checks the time and
17                 // returns once it has run for a second
18         printf("%s\n", str);
19     }
20     return 0;
    
```

September 26, 2018 TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.25

## VIRTUALIZING THE CPU - 3

```

prompt> gcc -o cpu cpu.c -Wall
prompt> ./cpu "A"
A
A
A
^C
prompt>
    
```

■ Runs forever, must Ctrl-C to halt...

September 26, 2018 TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.26

## VIRTUALIZATION THE CPU - 4

```

prompt> ./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D &
[1] 7353
[2] 7354
[3] 7355
[4] 7356
A
B
D
C
A
B
D
C
A
C
B
D
...
    
```

Even though we have only one processor all four instances of our program seem to be running at the same time!

September 26, 2018 TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.27

## VIRTUALIZING MEMORY

- Computer memory is treated as a large array of bytes
- Programs store all data in this large array
  - Read memory (load)
  - Specify an address to read data from
  - Write memory (store)
  - Specify data to write to an address

September 26, 2018 TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.28

## VIRTUALIZING MEMORY - 2

■ Program to read/write memory:

```

1  #include <unistd.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "common.h"
5
6  int
7  main(int argc, char *argv[])
8  {
9     int *p = malloc(sizeof(int)); // a1: allocate some
10                                     memory
11     assert(p != NULL);
12     printf("(%d) address of p: %08x\n",
13            getpid(), (unsigned) p); // a2: print out the
14                                     address of the memory
15     *p = 0; // a3: put zero into the first slot of the memory
16     while (1) {
17         Spin(1);
18         *p = *p + 1;
19         printf("(%d) p: %d\n", getpid(), *p); // a4
20     }
21     return 0;
    
```

September 26, 2018 TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.29

## VIRTUALIZING MEMORY - 3

■ Output of mem.c

```

prompt> ./mem
(2134) memory address of p: 00200000
(2134) p: 1
(2134) p: 2
(2134) p: 3
(2134) p: 4
(2134) p: 5
^C
    
```

- int value stored at 00200000
- program increments int value

September 26, 2018 TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.30

## VIRTUALIZING MEMORY - 4

- Multiple instances of mem.c

```

prompt> ./mem & ./mem &
[1] 24113
[2] 24114
(24113) memory address of p: 00200000
(24114) memory address of p: 00200000
(24113) p: 1
(24114) p: 1
(24114) p: 2
(24113) p: 2
(24114) p: 3
(24114) p: 3
...
    
```

- (int\*)p receives the same memory location 00200000
- Why does modifying (int\*)p in program #1 (PID=24113), not interfere with (int\*)p in program #2 (PID=24114) ?
  - The OS has "virtualized" memory, and provides a "virtual" address

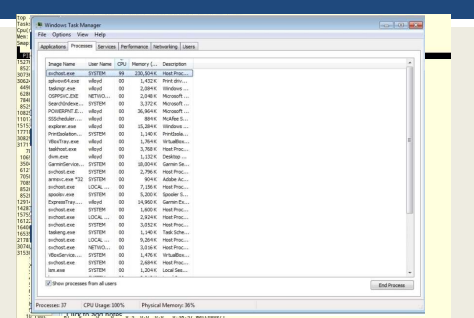
September 26, 2018    TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.31

## VIRTUAL MEMORY

- Key take-aways:
  - Each process (program) has its own **virtual address space**
  - The OS maps virtual **address spaces** onto **physical memory**
  - A memory reference from one process can not affect the address space of others.
    - Isolation**
  - Physical memory, a **shared resource**, is managed by the OS

September 26, 2018    TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.32

## CONCURRENCY



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It lists various system and user processes, including explorer.exe, chrome.exe, and several instances of java.exe. The bottom status bar indicates 37 processes, 100% CPU usage, and 36% physical memory usage.

- Linux: 654 tasks
- Windows: 37 processes
- The **OS** appears to run many programs at once, juggling them
- Modern **multi-threaded** programs feature concurrent threads and processes
- What is a key difference between a process and a thread?**

September 26, 2018    TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.33

## CONCURRENCY

- Linux: 654 tasks
- Windows: 37 processes
- The **OS** appears to run many programs at once, juggling them
- Modern **multi-threaded** programs feature concurrent threads and processes
- What is a key difference between a process and a thread?**

September 26, 2018    TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.34

## CONCURRENCY - 2

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "common.h"
4
5  volatile int counter = 0;
6  int loops;
7
8  void
9
10
11
12
13
14 }
15 ...
    
```

**Not the same as Java volatile:**  
 Provides a compiler hint that an object may change value unexpectedly (in this case by a separate thread) so aggressive optimization must be avoided.

thread.c  
 Listing continues ...

September 26, 2018    TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.35

## CONCURRENCY - 3

```

16  int
17  main(int argc, char *argv[])
18  {
19      if (argc != 2) {
20          fprintf(stderr, "usage: threads <value>\n");
21          exit(1);
22      }
23      loops = atoi(argv[1]);
24      pthread_t p1, p2;
25      printf("Initial value : %d\n", counter);
26
27      pthread_create(&p1, NULL, worker, NULL);
28      pthread_create(&p2, NULL, worker, NULL);
29      pthread_join(p1, NULL);
30      pthread_join(p2, NULL);
31      printf("Final value : %d\n", counter);
32      return 0;
33  }
    
```

- Program creates two threads
- Check documentation: "man pthread\_create"
- worker() method counts from 0 to argv[1] (loop)

September 26, 2018    TCCS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma    L1.36

**Linux "man" page example**

```

PTHREAD_CREATE(3)      Linux Programmer's Manual      PTHREAD_CREATE(3)

NAME
    pthread_create - create a new thread

SYNOPSIS
    #include <pthread.h>

    int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
        void *(*start_routine)(void *), void *arg);

    Compile and link with -pthread.

DESCRIPTION
    The pthread_create() function starts a new thread in the calling process. The new thread starts execution by invoking start_routine(); arg is passed as the sole argument of start_routine().

    The new thread terminates in one of the following ways:
    * It calls pthread_exit(), specifying an exit status value that is available to another thread in the same process that calls pthread_join().
    * It returns from start_routine(). This is equivalent to calling pthread_exit() with the value supplied in the return statement.
    * It is canceled (see pthread_cancel(3)).
    * Any of the threads in the process calls exit(3), or the main thread performs a return from main(). This causes the termination of all threads in the process.

    The attr argument points to a pthread_attr_t structure whose contents are used at thread creation time to determine attributes for the new thread; this structure is initialized using pthread_attr_t and related functions. If attr is NULL, then the thread is created with default attributes.
    
```

September 26, 2018    TCSS422: Operating Systems [Fall 2018]    L1.37  
 School of Engineering and Technology, University of Washington - Tacoma

## CONCURRENCY - 4

- Command line parameter argv[1] provides loop length
- Defines number of times the shared counter is incremented
- Loops: 1000


```

prompt> gcc -o thread thread.c -Wall -pthread
prompt> ./thread 1000
Initial value : 0
Final value : 2000
    
```

- Loops 100000

```

prompt> ./thread 100000
Initial value : 0
Final value : 143012 // huh??
prompt> ./thread 100000
Initial value : 0
Final value : 137298 // what the??
    
```



September 26, 2018    TCSS422: Operating Systems [Fall 2018]    L1.38  
 School of Engineering and Technology, University of Washington - Tacoma

## CONCURRENCY - 5

- When loop value is large why do we not achieve 200000 ?
- C code is translated to (3) assembly code operations
  - Load counter variable into register
  - Increment it
  - Store the register value back in memory
- These instructions happen concurrently and VERY FAST
- (P1 || P2) write incremented register values back to memory, While (P1 || P2) read same memory
- Memory access here is **unsynchronized (non-atomic)**
- Some of the increments are lost

September 26, 2018    TCSS422: Operating Systems [Fall 2018]    L1.39  
 School of Engineering and Technology, University of Washington - Tacoma

## W To perform parallel work, a single process may:

Launch multiple threads to execute code in parallel while sharing global data in memory	Launch multiple processes to execute code in parallel without sharing global data in memory	Both A and B	None of the above
---	---	--------------	-------------------

Start the presentation to see live content. Still no live content? Install the app or get help at [PallEx.com/app](http://PallEx.com/app)    Total Results

## PARALLEL PROGRAMMING

- To perform parallel work, a single process may:
- A. Launch multiple threads to execute code in parallel while sharing global data in memory
- B. Launch multiple processes to execute code in parallel without sharing global data in memory
- C. Both A and B
- D. None of the above

September 26, 2018    TCSS422: Operating Systems [Fall 2018]    L1.41  
 School of Engineering and Technology, University of Washington - Tacoma

## PERSISTENCE

- DRAM: Dynamic Random Access Memory: DIMMs/SIMMs
  - Stores data while power is present
  - When power is lost, data is lost (*volatile*)
- Operating System helps "persist" data more **permanently**
  - I/O device(s): hard disk drive (HDD), solid state drive (SSD)
  - File system(s): "catalog" data for storage and retrieval

September 26, 2018    TCSS422: Operating Systems [Fall 2018]    L1.42  
 School of Engineering and Technology, University of Washington - Tacoma

## PERSISTENCE - 2

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <assert.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6
7  int
8  main(int argc, char *argv[])
9  {
10     int fd = open("/tmp/file", O_WRONLY | O_CREAT
11                 | O_TRUNC, S_IRWXU);
12     assert(fd > -1);
13     int rc = write(fd, "hello world\n", 13);
14     assert(rc == 13);
15     close(fd);
16     return 0;
    
```

- `open()`, `write()`, `close()`: OS system calls for device I/O
- Note: man page for `open()`, `write()` require page number: "man 2 open", "man 2 write", "man close"

TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L2.43

## PERSISTENCE - 3

- To write to disk, OS must:
  - Determine where on disk data should reside
  - Perform sys calls to perform I/O:
    - Read/write to file system (*inode record*)
    - Read/write data to file
- Provide fault tolerance for system crashes
  - Journaling: Record disk operations in a journal for replay
  - Copy-on-write - replicating shared data - see ZFS
  - Carefully order writes on disk

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L2.44

## SUMMARY: OPERATING SYSTEM DESIGN GOALS


- **ABSTRACTING THE HARDWARE**
  - Makes programming code easier to write
  - Automate sharing resources – save programmer burden
- **PROVIDE HIGH PERFORMANCE**
  - Minimize overhead from OS abstraction (Virtualization of CPU, RAM, I/O)
  - Share resources fairly
  - Attempt to tradeoff performance vs. fairness → consider priority
- **PROVIDE ISOLATION**
  - User programs can't interfere with each other's virtual machines, the underlying OS, or the sharing of resources

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L2.45

## SUMMARY: OPERATING SYSTEM DESIGN GOALS - 2


- **RELIABILITY**
  - OS must not crash, 24/7 Up-time
  - Poor user programs must not bring down the system:

Blue Screen


- Other Issues:
  - Energy-efficiency
  - Security (of data)
  - Cloud: Virtual Machines


September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L2.46

# QUESTIONS



## TCSS 422 – Fall 2018 Special features!

- **Going green...**
  - 20% reduction of carbon footprint
- 16 in person class meetings
  - Online lectures: Monday April 16, Wednesday April 18
  - No class: Monday April 9, Monday May 28
- Saves commuting time
  - Less fuel expenses
- Easier to achieve perfect attendance
- Final exam Monday June 4<sup>th</sup>
  - 71 days from now...



TCSS 422  
FALL 2018

September 26, 2018 TCSS422: Operating Systems [Fall 2018]  
 School of Engineering and Technology, University of Washington - Tacoma L1.48