TCSS 422: OPERATING SYSTEMS

INTRODUCTION

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

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma

1

TCSS 422 – Spring 2020

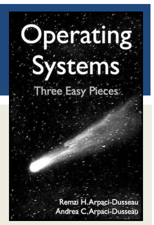
TCSS 422 SPRING 2020

- ***Online is green…***
  - **100% reduction of carbon footprint from transit**
- Saves commuting time
  - Less fuel expenses
- Easier to achieve perfect attendance – *all lectures streamed LIVE, recorded for 24/7 availability*
- 20 class meetings
  - 1 Monday holiday in Spring: May 25
- Final exam Tuesday June 9th

L1.2

2

OBJECTIVES



- Syllabus, Course Introduction
- C Review
- Background 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

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.3

3

SILVER LINING

- Practice use of technology for remote collaborative work
- Professor conducted MS research at VA Tech on distributed remote work in early 2000s
- Computer Science is a unique field where you can work in a job entirely remotely from home or from any location
- Colleague from undergrad, Scott Teresi, MS in CS from Univ of Illinois – works for British company remotely for over a decade
  - Well paid!
  - Never physically met boss until recently when company bought
  - Now makes occasional trips to the UK

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.4

4

RESOURCES FOR SPRING 2020

- Free internet access from Comcast (?):
- https://www.thenewstribune.com/news/business/article241188606.html
- UW Tacoma Information Technology - Laptops for loan:
- https://www.tacoma.uw.edu/information-technology/equipment-checkout
- UW Tacoma Library – Laptops for loan:
- https://www.tacoma.uw.edu/learning-research-commons/laptops-available-checkout
- Textbook coupon 20% off “LULU20” until Thursday at 11:59pm
- http://www.lulu.com/shop/remzi-arpaci-dusseau-and-andrea-arpaci-dusseau/operating-systems-three-easy-pieces-softcover-version-100/paperback/product-23779877.html

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.5

5

TCSS422 – SPRING 2020 COMPUTER OPERATING SYSTEMS

- Syllabus
- Grading
- Schedule
- Assignments

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.6

6

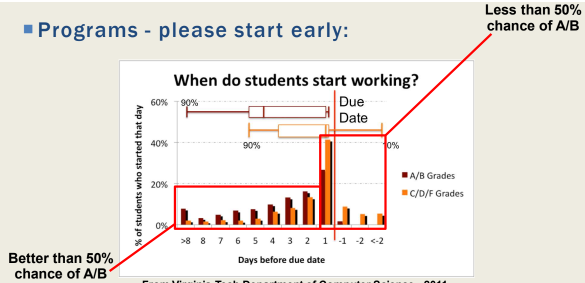
TCS422 COURSE WORK

- **Assignments (45%)**
  - 4 Assignments: roughly every two weeks
  - Submit ALL programming assignments via Canvas – no email
    - Email submissions are prone to be lost
- **Tutorials/Quizzes/In-class activities (15%)**
  - ~ 5-6 quizzes
  - Drop lowest two
  - Variety of formats: collaborative in class (*via Zoom breakout rooms*), online, reading, tutorial
- **Exams: Midterm and Final (40%)**
  - Online via the Canvas system
  - Final exam is comprehensive, with emphasis on new material

7

TCSS 422: PROGRAM DUE DATES

- Programs - please start early:



From Virginia Tech Department of Computer Science - 2011

8

TCSS 422: PROGRAMS

- *Tentative - subject to change*
- **Assignment 0:** Introduction to Linux, Ubuntu Virtual Machine
- **Assignment 1:** Programming with multiple processes (in C)
- **Assignment 2:** Multithreaded programming and concurrency (C or Java)
- **Assignment 3:** Kernel (real) mode programming (in C)

9

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 CSS lab mentors (*checking on availability for Spring 2020*)
    - If less familiar with C/pointers (TCSS 333/380), ***BUDGET MORE TIME***

10

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

11

UBUNTU 18.04 – VIRTUAL MACHINE INSTALLATION

- Ubuntu 18.04 on Oracle VirtualBox
- HOW-TO installation videos:
- Windows 10
- https://www.youtube.com/watch?v=QbmRXJJKsvs
- Mac OS X (not specific to 18.04)
- https://www.youtube.com/watch?v=sNixOS6mHIU
- > AFTER VirtualBox, INSTALL THE **Guest Additions**
  - IMPORTANT USABILITY ADD-ON: Provides file system sharing, clipboard integration, mouse tricks
- https://www.youtube.com/watch?v=qNecdUsuTPw

12

C PROGRAMING IN TCSS 422

- Many OSes are coded primarily in C and Assembly Language
- C is a particularly useful language for working with hardware / hardware drivers and operating systems
- C allows writing programs that can directly access the computer's physical memory (in kernel/real mode) providing nearly the power and speed of assembly language
  - *But in a much easier to write high-level language*
- Ideally, all university operating system courses are taught in C/C++. Our textbook is in C/C++
  - *This quarter we will offer the option of assignment of completing assignment 2 in Java (multithreaded programming)*

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.13

13

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
- Spring quarter hours will be posted – if available

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.14

14

INSTRUCTOR HELP

- Office hours: tentative 3:30-4:30p TR after class
  - May change based on course survey results
  - Also available by appointment
- Take ***ownership*** of your educational outcome
  - 10 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 on zoom !! *****
  - Also questions after class, email, Canvas discussion boards
  - Seek help using UWT resources, the internet, YouTube videos (video.google.com) and online tutorials

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.15

15

CLASS PARTICIPATION

- **Questions and discussion are strongly encouraged**
  - Leverage your educational investment
  - All questions are encouraged!
  - This instructor appreciates questions at all levels – there is no judgement for any question
- **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
- **Poll-EV**

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.16

16

WE WILL RETURN AT 2:40PM

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - coma | L1.17

17

C REVIEW SURVEY

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - coma | L1.18

18

DEMOGRAPHICS SURVEY

SEE LINK AT:
http://faculty.washington.edu/wlloyd/courses/tcss422/announcements.html

19

INTRODUCTION TO OPERATING SYSTEMS

20

VIRTUAL MACHINE SURVEY

- Please complete the Virtual Machine Survey to request a "School of Engineering and Technology" remote hosted Ubuntu VM
- https://forms.gle/R8N4HTjx6qKf1VJ88

21

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

22

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.

23

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
    - …

24

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

25

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

26

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

27

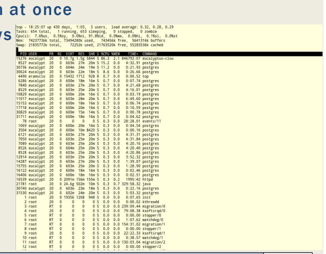
ABSTRACTION CHALLENGES

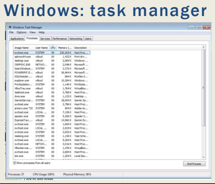
- What level of abstraction?
  - How much of the underlying hardware should be exposed?
    - What if **too much**?
    - What if **too little**?
- What are the correct abstractions?
  - Security concerns

28

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





29

VIRTUALIZING THE CPU - 2

- Simple Looping C Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <assert.h>
#include "common.h"

int
main(int argc, char *argv[])
{
        if (argc != 2) {
                fprintf(stderr, "usage: cpu <string>\n");
                exit(1);
        }
        char *str = argv[1];
        while (1) {
                Spin(1); // Repeatedly checks the time and
                returns once it has run for a second
                printf("%s\n", str);
        }
        return 0;
}
```


30

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…

31

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!

32

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

33

VIRTUALIZING MEMORY - 2

- Program to read/write memory:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include "common.h"

int
main(int argc, char *argv[])
{
        int *p = malloc(sizeof(int));  // a1: allocate some
                                          memory
        assert(p != NULL);
        printf("(%d) address of p: %08x\n",
                getpid(), (unsigned) p);  // a2: print out the
                                          address of the memmory
        *p = 0;  // a3: put zero into the first slot of the memory
        while (1) {
                Spin(1);
                *p = *p + 1;
                printf("(%d) p: %d\n", getpid(), *p); // a4
        }
        return 0;
}
```


34

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

35

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
(24113) 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

36

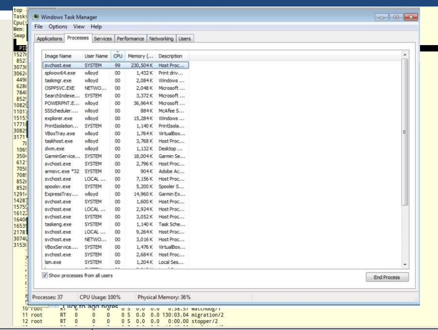
VIRTUAL MEMORY

- 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 <u>shared resource</u>, is managed by the OS

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.37

37

CONCURRENCY



March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.38

38

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
- ***<u>What is a key difference between a process and a thread?</u>***

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.39

39

CONCURRENCY - 2

```
#include <stdio.h>
#include <stdlib.h>
#include "common.h"

volatile int counter = 0;
int loops;

void
...
```

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

thread.c

Listing continues …

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.40

40

CONCURRENCY - 3

```
int
main(int argc, char *argv[])
{
        if (argc != 2) {
                fprintf(stderr, "usage: threads <value>\n");
                exit(1);
        }
        loops = atoi(argv[1]);
        pthread_t p1, p2;
        printf("Initial value : %d\n", counter);

        Pthread_create(&p1, NULL, worker, NULL);
        Pthread_create(&p2, NULL, worker, NULL);
        Pthread_join(p1, NULL);
        Pthread_join(p2, NULL);
        printf("Final value : %d\n", counter);
        return 0;
}
```

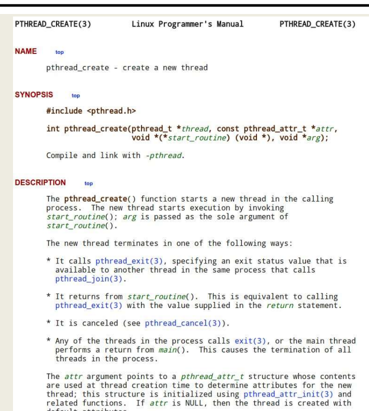
- Program creates two threads
- Check documentation: “man pthread_create”
- worker() method counts from 0 to argv[1] (loop)

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.41

41

Linux “man” page

<u>example</u>



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

NAME top

pthread_create - create a new thread

SYNOPSIS top

#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 top

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(3), specifying an exit status value that is available to another thread in the same process that calls pthread_join(3).

* It returns from *start_routine*(). This is equivalent to calling pthread_exit(3) 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_init(3) and related functions. If *attr* is NULL, then the thread is created with default attributes.

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.42

42

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??
```

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.43

43

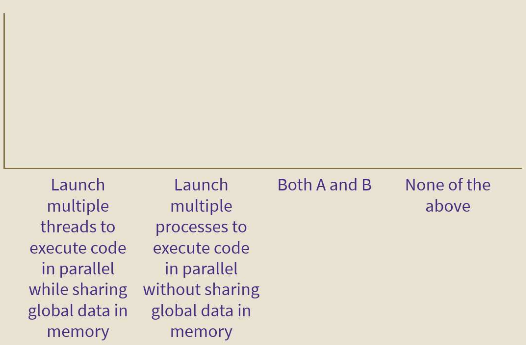
CONCURRENCY - 5

- When loop value is large why do we not achieve 200000 ?
- C code is translated to (3) assembly code operations
1. Load counter variable into register
2. Increment it
3. 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*

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.44

44

To perform parallel work, a single process may:



Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app

45

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

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L1.46

46

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

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.47

47

PERSISTENCE - 2

```
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <fcntl.h>
#include <sys/types.h>

int
main(int argc, char *argv[])
{
        int fd = open("/tmp/file", O_WRONLY | O_CREAT
                      | O_TRUNC, S_IRWXU);
        assert(fd > -1);
        int rc = write(fd, "hello world\n", 13);
        assert(rc == 13);
        close(fd);
        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"`

March 31, 2020 | TCSS422: Operating Systems [Spring 2020] School of Engineering and Technology, University of Washington - Tacoma | L2.48

48

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

49

SUMMARY: OPERATING SYSTEM DESIGN GOALS

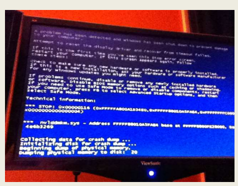
- **ABSTRACTING THE HARDWARE**
  - Makes programming code easier to write
  - Automate sharing resources – save programmer burden
- **PROVIDE HIGH PERFORMANCE**
  - Minimize overhead from OS abstraction (Virtualization of CPU, RAM, I/O)
  - Share resources fairly
  - Attempt to tradeoff performance vs.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

50

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

51

QUESTIONS

52