# TCSS 422: OPERATING SYSTEMS

## INTRODUCTION

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

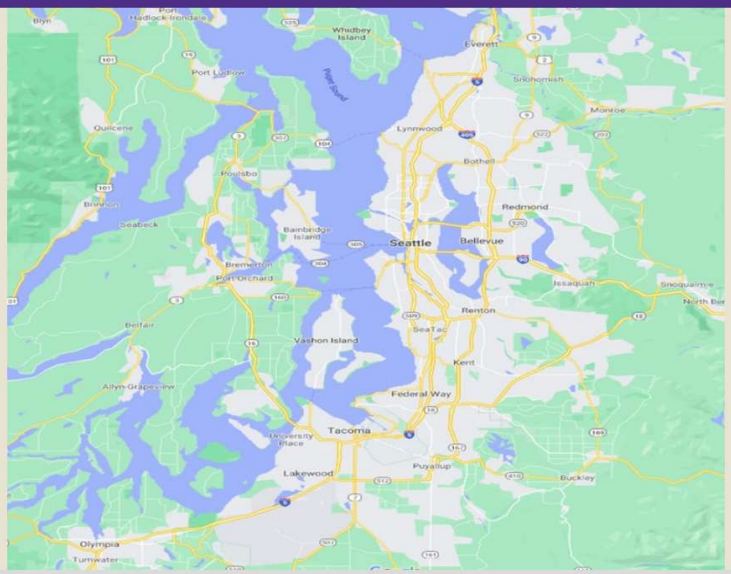
March 30, 2021

TCSS422: Operating Systems [Spring 2021]
School of Engineering and Technology, University of Washington Tacoma

## Where are you joining us from? (WORLD VERSION)

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

## Where are you joining us from? (PUGET SOUND REGION)




Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

## INTRODUCTIONS: What is your name? nickname / alias? and list one or more areas of interest in Computer Science:

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

## OBJECTIVES – 3/30


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



## TCSS 422 – Spring 2021


- ***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 31
- Final exam Thursday June 10th

TCSS 422 SPRING 2021

L1.6

## SILVER LINING FOR ONLINE LEARNING

- ONLINE LEARNING: practice use of technology for remote collaborative work
- Professor conducted Masters thesis 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 company was recently bought
  - Before covid – now making occasional trips to the UK



## RESOURCES FOR SPRING 2021

- **Low Cost or Free Technology Services from UW Tacoma:**
- https://www.tacoma.uw.edu/about-uw-tacoma/low-cost-or-free-technology-services
- **Resources for students during the Coronavirus pandemic:**
- https://www.tacoma.uw.edu/chancellor/resources-students-during-coronavirus-pandemic
  https://www.tacoma.uw.edu/uwt/it/it-resources-telework-and-attending-class-remotely
- **UW Tacoma Information Technology & Library Laptops for loan:**
- https://itconnect.uw.edu/work/working-remotely/technology-for-working-remotely/acquiring-computers-and-hardware-for-working-remotely/
- https://www.tacoma.uw.edu/learning-research-commons/laptops-available-checkout

## RESOURCES - 2

- **Textbook coupon 10% off "WRITER10" until Friday 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**



## TCSS422 – SPRING 2021 COMPUTER OPERATING SYSTEMS

- Syllabus
- Grading
- Schedule
- Assignments

**See website at:**
**http://faculty.washington.edu/wlloyd/courses/tcss422**

**Enables access using mobile device w/o logging into Canvas**

**Website also integrated into Canvas**

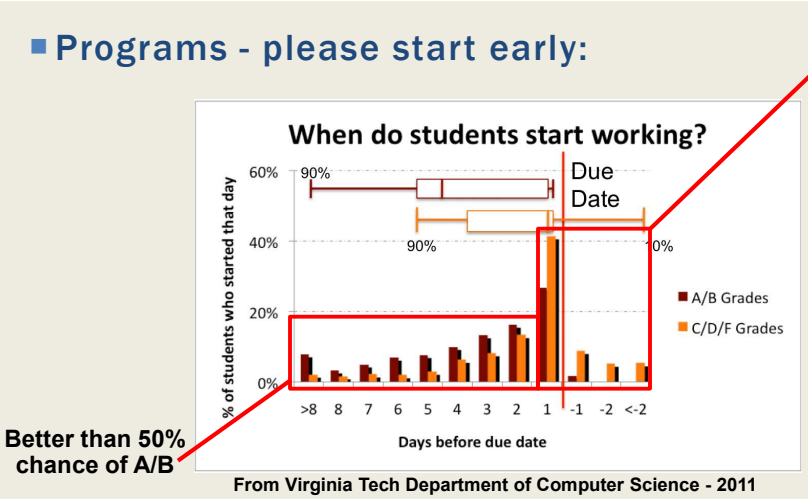
# TCS422 COURSE WORK

- **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



# TCSS 422: PROGRAM DUE DATES

- Programs - please start early:




From Virginia Tech Department of Computer Science - 2011

## 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)



## 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 2021*)
    - If less familiar with C/pointers (TCSS 333/380), ***BUDGET MORE TIME***

## UBUNTU 20.04 – VIRTUAL MACHINE

- Ubuntu 20.04
  - 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



## UBUNTU 20.04 – VIRTUAL MACHINE INSTALLATION

- Ubuntu 20.04 on Oracle VirtualBox
- HOW-TO installation videos:
- Windows 10
- **https://youtu.be/x3Zpe1rIPFE**
- Mac OS X
- **https://youtu.be/Hzji7w882OY**
- > AFTER VirtualBox, INSTALL THE **Guest Additions**
  - IMPORTANT USABILITY ADD-ON: Provides file system sharing, clipboard integration, mouse tricks
- **https://youtu.be/Kbez-XdXqrw**

## 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)*



## C MENTORING

- https://www.tacoma.uw.edu/institute-technology/student-support-workshops-mentors
- School of Engineering and Technology Mentors
- Office hours held online via Zoom
- Varied hours and availability based on mentors schedules
- Monday – Thursday: ~9:30 am – 9:00 pm
- Friday: ~ 9:30 - 3:30 pm
- Spring quarter hours will be posted once available
- Student mentors managed by SET Monika Sobolewska

## INSTRUCTOR HELP

- Office hours: tentative 5:40p TR after class
  - Additional hours based on 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



## CLASS PARTICIPATION

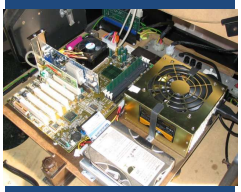
- Questions and discussion are strongly encouraged
  - Leverage your educational investment
  - All questions are encouraged!
  - This instructor appreciates questions at all levels – there is no judgement for any question
- Daily feedback surveys
  - How much is new vs. review?
  - Checking the pace…
  - 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

## OBJECTIVES – 3/30

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



## C REVIEW SURVEY



# OBJECTIVES – 3/30

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

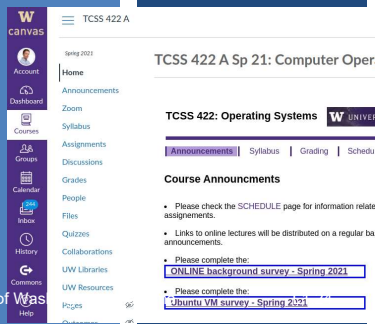


# BACKGROUND & VM SURVEYS

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

or in *Canvas* under "**Announcements**"

we will resume at ~4:50pm



# WE WILL RETURN AT 4:50PM



## STUDENT BACKGROUND SURVEY

- Please complete the Student Background Survey
- https://forms.gle/yr6Dc9x9rX516U6t6

## VIRTUAL MACHINE SURVEY

- Please complete the Virtual Machine Survey to request a "School of Engineering and Technology" remote hosted Ubuntu VM
- https://forms.gle/BR2G1wr9RDBVB9AK8



## OBJECTIVES – 3/30

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

# INTRODUCTION TO OPERATING SYSTEMS



## 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

## 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.**



## OBJECTIVES – 3/30

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

## 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
    - …



## OBJECTIVES – 3/30

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

## 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



## 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

## 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



## 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

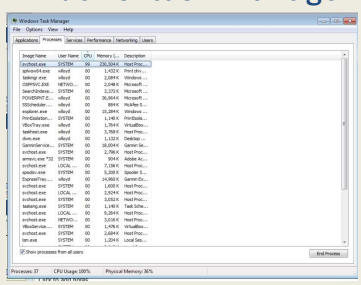
# OBJECTIVES – 3/30

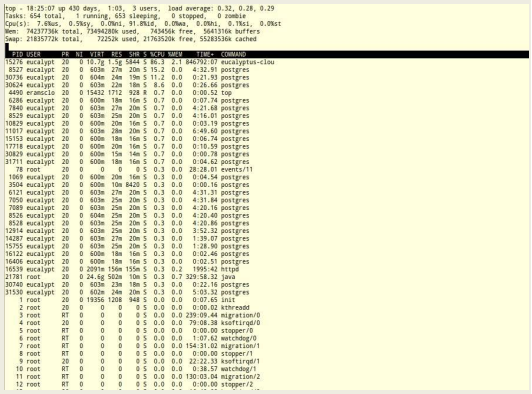
- Syllabus, Course Introduction
- C Review Survey
- Background Survey
- Chapter 4: Operating Systems – Three Easy Pieces
  - Introduction to operating systems
  - Management of resources
  - Concepts of virtualization/abstraction
  - **Three Easy Pieces: CPU**, Memory, I/O
  - Concurrency
  - Operating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



# 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





# 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
                        returns once it has run for a second
17                      printf("%s\n", str);
18              }
19              return 0;
20      }
```



# 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…

# 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!**



# OBJECTIVES – 3/30

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

# 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



# 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
                                                  memory
10              assert(p != NULL);
11              printf("(%d) address of p: %08x\n",
12                      getpid(), (unsigned) p);  // a2: print out the
                                                     address of the memmory
13              *p = 0;  // a3: put zero into the first slot of the memory
14              while (1) {
15                      Spin(1);
16                      *p = *p + 1;
17                      printf("(%d) p: %d\n", getpid(), *p); // a4
18              }
19              return 0;
20      }
```

# 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



# 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

## 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



## OBJECTIVES – 3/30

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

# 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



# 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
                                | O_TRUNC, S_IRWXU);
11              assert(fd > -1);
12              int rc = write(fd, "hello world\n", 13);
13              assert(rc == 13);
14              close(fd);
15              return 0;
16      }
```

- 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"

## PERSISTENCE - 3

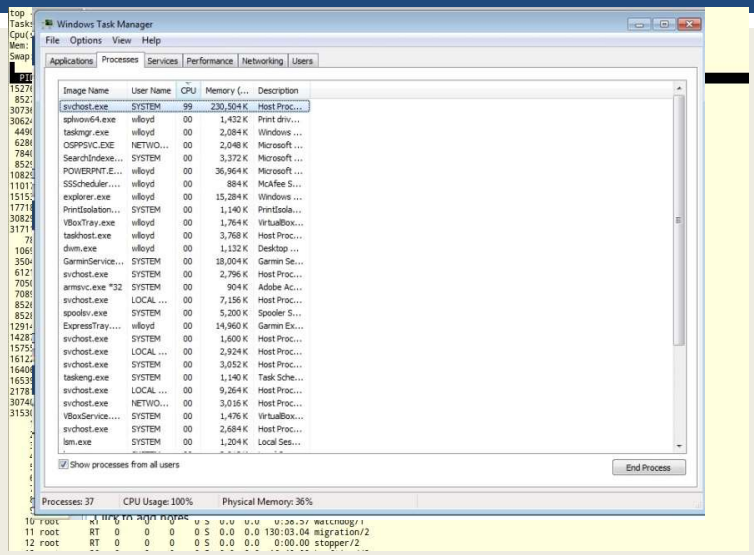
- To write to disk, OS must:
  - Determine where on disk data should reside
  - Perform sys calls to perform I/O:
    - Read/write to file system (*inode record*)
    - Read/write data to file
- Provide fault tolerance for system crashes
  - Journaling: Record disk operations in a journal for replay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



## OBJECTIVES – 3/30

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

# CONCURRENCY






# 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?***

# 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 than an object may change value unexpectedly (in this case by a separate thread) so aggressive optimization must be avoided.***

**thread.c**

**Listing continues …**



# 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)**

**Linux "man" page**

**example**

```
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.
```



# 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??
```

# CONCURRENCY - 5

- When loop value is large why do we not achieve 200000 ?

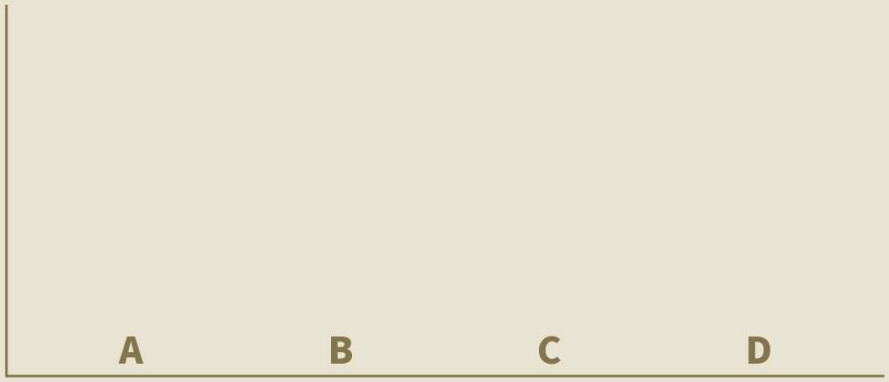
- C code is translated to (3) assembly code operations

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*



## 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 while sharing global data in memory

C: Both A and B

D: None of the above

## 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



## OBJECTIVES – 3/30

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

## 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



## 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

# QUESTIONS