


TCCS 422: OPERATING SYSTEMS

INTRODUCTION

Wes J. Lloyd

Institute of Technology

University of Washington - Tacoma



January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.1

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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.2

TCCS422 – WINTER 2018
COMPUTER OPERATING SYSTEMS

Syllabus

Grading

Schedule

Assignments

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma


L1.3

TCCS 422 – Winter 2018
Special features!

Going green...

- 15% reduction of carbon footprint

- Just 17 class meetings
- In contrast to usual 20
- 3 Monday holidays:
Jan 1, Jan 15, Feb 19
- Saves commuting time
- Less fuel expenses
- Easier to achieve perfect attendance
- Final exam Monday March 12th
- Just 68 days from now... (9.7 weeks)



TCCS 422
WINTER
2018

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.4

TCS422 COURSE WORK

Assignments

- Assignment 0: Linux /scripting
- Assignments 1 – 3 ~4: roughly every two weeks

Quizzes

- ~ 7-8 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

January 3, 2018

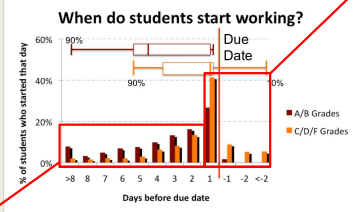
TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.5

TCCS 422: PROGRAM DUE DATES

Programs - please start early:

When do students start working?



From Virginia Tech Department of Computer Science - 2011

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.6

Slides by Wes J. Lloyd

L1.1

TCCS 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 (TCCS 333),
BUDGET MORE TIME
- New this quarter - 5% bonus for submitting **by the originally posted due date**
 - Excludes any class-wide extensions

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.7

UBUNTU 16.04 – VIRTUAL MACHINE

■ Ubuntu 16.04

- 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.8

UBUNTU 16.04 – VIRTUAL MACHINE INSTALLATION

■ Ubuntu 16.04

■ Search online for YouTube videos, tutorials

■ Search how to install the “Guest Additions”

- Provides file system sharing, clipboard integration, mouse tricks

■ Windows 10

■ <https://www.youtube.com/watch?v=DPIPC25xzUM>

■ Mac OS X

■ <https://www.youtube.com/watch?v=sNixOS6mHIU>

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.9

C PROGRAMING IN TCCS 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%

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.10

C MENTORING

■ <https://www.tacoma.uw.edu/institute-technology/student-support-workshops-mentors>

- Institute of Technology Mentors
- Located in Science 106 / 108 Labs
- Monday – Thursday: ~10 am – 7 pm
- Friday: ~ 12-5pm
- Winter quarter hours to be posted

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.11

INSTRUCTOR HELP

■ Office hours: to be announced, and by appointment

■ End of class: good for quick questions, assignment Q&A

■ It will be difficult to tutor **all** 70 students individually on C

■ Take **ownership** of your educational outcome

- Time spent in TCCS 422 is just ~0.4% of an IT career
- Make the most of this **limited** time
 - 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.12

CLASS PARTICIPATION

- **Questions and discussion are strongly encouraged**
 - Leverage your educational investment
 - All questions are encouraged! All are good!
 - Better to ask redundant questions, than to be unsure!
- **Daily feedback surveys**
 - Helpful to know if topics are not clear
 - Use the survey to write questions that come to you during the lecture
- **Poll-EV to be Introduced**

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.13

C REVIEW




January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.14

DEMOGRAPHICS SURVEY

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




January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.15

INTRODUCTION TO OPERATING SYSTEMS



January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.16

OBJECTIVES

- 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

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of 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.

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of 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
 - ...

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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

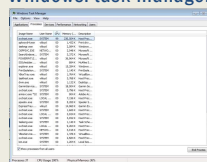
January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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



January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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;
}
```

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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...

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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 of programs seem to be running at the same time!

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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("(kd) 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("(kd) p: %d\n", getpid(), *p); // a4
20     }
21     return 0;
}
```

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of 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

January 3, 2018

TCCS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.30

- 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

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.31

- 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

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.32

[illegible]

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.33

- 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?**

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.34

```

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      Provides a compile
12      unexpectedly (in th
13      optimization must
14
15

```

Listing continues ...

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.35

```

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)

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.36

Linux
"man"
page

example

PTHREAD_CREATE(3)Linux Programmer's ManualPTHREAD_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()`).

Any of the threads in the process calls `exit()`, 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()` and related functions. If `attr` is `NULL`, then the thread is created with default attributes.

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.37


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??
```



January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.38

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

January 3, 2018

TCSS422: Operating Systems [Winter 2018]
Institute of Technology, University of Washington - Tacoma

L1.39

QUESTIONS

