


# TCSS 422: OPERATING SYSTEMS

## INTRODUCTION



Wes J. Lloyd  
Institute of Technology  
University of Washington - Tacoma

September 28, 2016 TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

## OBJECTIVES

- Introduce operating systems
- Management of resources
- Concepts of virtualization/abstraction
  - CPU, Memory, I/O
- Operating system design goals

September 28, 2016 TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma L1.2

## 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 28, 2016 TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma L1.3

## 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 28, 2016 TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma L1.4

## 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 28, 2016 TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma L1.5

## 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 28, 2016 TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma L1.6

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.7

## 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

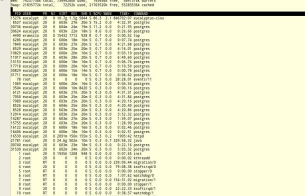
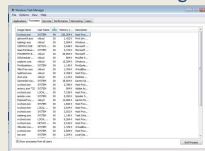
September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.8

## 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



September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.9

## 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;
21 }
```

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.10

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.11

## 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!

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.12

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.13

## 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("(1) address of p: %08x\n",
13           getpid(), (unsigned) p); // a2: print out the
14    // address of the memory
15    while (1) {
16        *p = 0; // a3: put zero into the first slot of the memory
17        Spin(1);
18        *p = *p + 1;
19        printf("(2) p: %d\n", getpid(), *p); // a4
20    }
21    return 0;
22 }
```

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.14

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.15

## 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
(24113) p: 2
(24114) 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

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.16

## VIRTUAL MEMORY

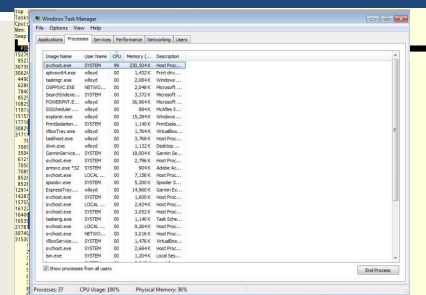
- Key take-aways:
  - Each process (program) has its own **virtual address space**
  - The OS maps virtual **address spaces** onto **physical memory**
  - A memory reference from one process can not affect the address space of others.
    - Isolation**
  - Physical memory, a **shared resource**, is managed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

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.17

## CONCURRENCY



September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.18

## 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 processes and threads?

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.19

## 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.*

Listing continues ...

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.20

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.21

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\_init(3) and related functions. If attr is NULL, then the thread is created with default attributes.

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.22

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.23

## 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

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.24

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.25

## 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
8  int
9  main(int argc, char *argv[])
10 {
11     int fd = open("/tmp/file", O_WRONLY | O_CREAT
12                  | O_TRUNC, S_IRWXU);
13     assert(fd > -1);
14     int rc = write(fd, "hello world\n", 13);
15     assert(rc == 13);
16     close(fd);
17     return 0;
18 }
```

- `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 2016]  
Institute of Technology, University of Washington - Tacoma

L1.26

## PERSISTENCE - 3

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

September 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.27

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.28

## 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 28, 2016

TCSS422: Operating Systems [Fall 2016]  
Institute of Technology, University of Washington - Tacoma

L1.29

## QUESTIONS

