


TCSS 422: OPERATING SYSTEMS

Processes & The Process API



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

October 7, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington Tacoma

1

OBJECTIVES – 10/7


- **Questions from 10/5**
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0
- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L3.2

2

VIRTUAL MACHINE SUPPORT ON APPLE M1

- Installing a Ubuntu Virtual Machine on Apple M1 MacBooks:
- FREE
- <https://mac.getutm.app/>
- MACs have switched to using ARM-based CPUs
 - Motivation: faster, less expensive than Intel-based CPUs



October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.3
-----------------	---	------

3

TEXT BOOK COUPON

- 15% off textbook code: **TRICK15** (*through Friday Oct 8*)
- <https://www.lulu.com/shop/remzi-arpaci-dusseau-and-andrea-arpaci-dusseau/operating-systems-three-easy-pieces-softcover-version-100/paperback/product-23779877.html?page=1&pageSize=4>

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.4
-----------------	---	------

4

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys **ON TIME**
- Tuesday surveys: due by ~ Wed @ 9p, closes 11:59p
- Thursday surveys: due ~ Mon @ 9p, closes 11:59p

TCSS 422 A > Assignments

Spring 2021

Search for Assignment

Home

Announcements

Zoom

Syllabus

Assignments

Discussions

Upcoming Assignments

TCSS 422 - Online Daily Feedback Survey - 4/1
Available until Apr 5 at 11:59pm | Due Apr 5 at 10pm | -1/1 pts

Quiz 0 - Background survey

October 7, 2021 TCSS422: Computer Operating Systems [Fall 2021] L3.5
School of Engineering and Technology, University of Washington - Tacoma

5

TCSS 422 - Online Daily Feedback Survey - 4/1

Quiz Instructions

Question 1 0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

1 2 3 4 5 6 7 8 9 10

Mostly Review To Me Equal New and Review Mostly New to Me

Question 2 0.5 pts

Please rate the pace of today's class:

1 2 3 4 5 6 7 8 9 10


Slow Just Right Fast

October 7, 2021 TCSS422: Computer Operating Systems [Fall 2021] L3.6
School of Engineering and Technology, University of Washington - Tacoma

6

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (28 respondents):
 - 1-mostly review, 5-equal new/review, 10-mostly new
 - **Average – 7.00 (↑ - previous 5.64)**
- Please rate the pace of today's class:
 - 1-slow, 5-just right, 10-fast
 - **Average – 5.89 (↑ - previous 5.38)**



October 7, 2021	TCSS422: Computer Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.7
-----------------	--	------

7

FEEDBACK

- **Trying to conceptualize the reason for virtual addresses (In Operating Systems)**
 - **Security:** if physical addresses were exposed, an attacker could acquire the physical address and attempt to read, modify, write the data
 - **Program Relocation:** because users only see virtual addresses, the OS can physically move programs to new locations without changing any user pointers
 - **Memory defragmentation:** OS can dynamically reorganize memory for better efficiency. All user pointers are virtual. Virtual pointers still work and are translated to new addresses
 - **Shared Libraries:** Two programs can have a virtual address (pointer) to a shared library that is mapped by the OS to a single physical address. The sharing and library location are abstracted. Shared libraries are important to save memory.

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.8
-----------------	---	------

8

MOTIVATION FOR LINUX

- It is worth noting the importance of Linux for today's developers and computer scientists.
- The CLOUD runs many virtual machines, recently in 2019 a key milestone was reached.
- Even on Microsoft Azure (the Microsoft Cloud), there were more Linux Virtual Machines (> 50%) than Windows.
- <https://www.zdnet.com/article/microsoft-developer-reveals-linux-is-now-more-used-on-azure-than-windows-server/>
- <https://www.zdnet.com/article/it-runs-on-the-cloud-and-the-cloud-runs-on-linux-any-questions/>
- The majority of application back-ends (server-side), cloud or not, run on Linux.
- This is due to licensing costs, example:

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.9
-----------------	---	------

9

MOTIVATION FOR LINUX - 2

- Consider an example where you're asked to develop a web services backend that requires 10 x 8-CPU-core virtual servers
- Your organization investigates hosting costs on Amazon cloud
- 8-core VM is "c5d.2xlarge"

Name	Instance type	Memory	vCPUs	Linux On Demand cost	Windows On Demand cost
C5 High-CPU Extra Large	c5d.xlarge	8.0 GiB	4 vCPUs	\$0.192000 hourly	\$0.376000 hourly
C5 High-CPU 18xlarge	c5d.18xlarge	144.0 GiB	72 vCPUs	\$3.456000 hourly	\$6.768000 hourly
C5 High-CPU Large	c5d.large	4.0 GiB	2 vCPUs	\$0.096000 hourly	\$0.188000 hourly
C5 High-CPU 24xlarge	c5d.24xlarge	192.0 GiB	96 vCPUs	\$4.608000 hourly	\$9.024000 hourly
C5 High-CPU Quadruple Extra Large	c5d.4xlarge	32.0 GiB	16 vCPUs	\$0.768000 hourly	\$1.504000 hourly
C5 High-CPU Metal	c5d.metal	102.0 GiB	66 vCPUs	\$4.608000 hourly	\$9.024000 hourly
C5 High-CPU Double Extra Large	c5d.2xlarge	16.0 GiB	8 vCPUs	\$0.384000 hourly	\$0.752000 hourly
C5 High-CPU 12xlarge	c5d.12xlarge	96.0 GiB	48 vCPUs	\$2.304000 hourly	\$4.512000 hourly
C5 High-CPU 9xlarge	c5d.9xlarge	72.0 GiB	36 vCPUs	\$1.728000 hourly	\$3.384000 hourly

- Windows hourly price 75.2¢
- Linux hourly price 38.4¢
- See: <https://www.ec2instances.info/>

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.10
-----------------	---	-------

10

MOTIVATION FOR LINUX - 2

One year cloud hosting cost:

- **WINDOWS**
10 VMs x 8,760 hours x \$.752 = \$65,875.20
- **Linux**
10 VMs x 8,760 hours x \$.384 = \$33,638.40

Windows comes at a 95.8% price premium

See: <https://www.ec2instances.info/>

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.11
-----------------	---	-------

11

OBJECTIVES - 10/7

- Questions from 10/5
- **C Review Survey - Due Sunday Oct 10**
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0
- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.12
-----------------	---	-------

12

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- **Student Background Survey**
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0

- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.13
-----------------	---	-------

13

STUDENT BACKGROUND SURVEY

- Please complete the Student Background Survey

- <https://forms.gle/BuJwXPwZpqf6cnTQ9>

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.14
-----------------	---	-------

14

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- **Virtual Machine Survey: VM requests sent to S. Rondeau**
- Assignment 0

- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.15
-----------------	---	-------

15

VIRTUAL MACHINE SURVEY

- Please complete the Virtual Machine Survey to request a “School of Engineering and Technology” remote hosted Ubuntu VM

- <https://forms.gle/V2sg4iW1awvhFx4W8>

- **Will close Thursday 10/7...**
- VM requests will be sent to Stephen Rondeau for set up

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.16
-----------------	---	-------

16

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- **Assignment 0**

- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.17
-----------------	---	-------

17

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0

- **Chapter 4: Processes**
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()


October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.18
-----------------	---	-------

18

CHAPTER 4: PROCESSES

Process State

```
graph TD
    created((created)) -- admitted --> ready((ready))
    ready -- scheduler dispatch --> running((running))
    running -- interrupt --> ready
    running -- I/O or event wait --> waiting((waiting))
    waiting -- I/O or event completion --> ready
    running -- exit --> terminated((terminated))
```


`/proc`

October 7, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L3.19

19

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0

- Chapter 4: Processes
 - **Process states, context switches**
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.20
-----------------	---	-------

20

PROCESS STATES

- **RUNNING**
 - Currently executing instructions

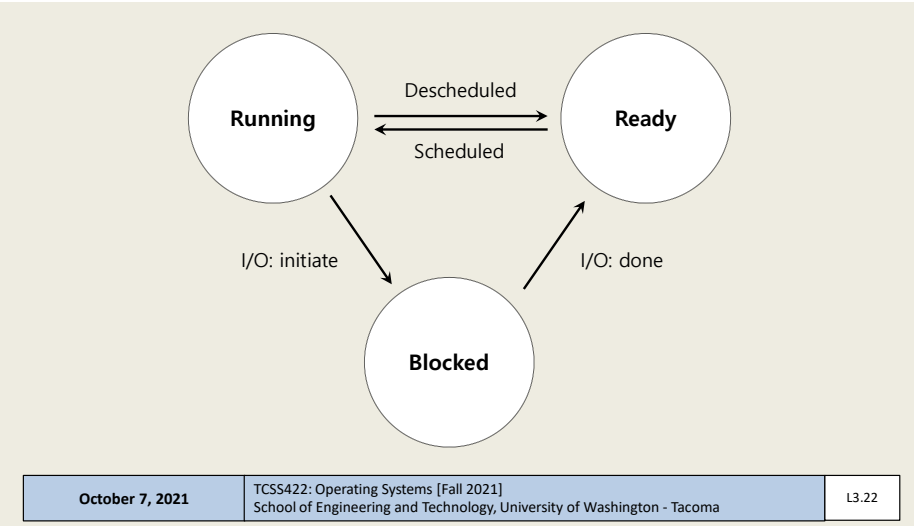
- **READY**
 - Process is ready to run, but has been preempted
 - CPU is presently allocated for other tasks

- **BLOCKED**
 - Process is **not** ready to run. It is waiting for another event to complete:
 - Process has already been initialized and run for awhile
 - Is now waiting on I/O from disk(s) or other devices

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.21
-----------------	---	-------

21

PROCESS STATE TRANSITIONS



October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.22
-----------------	---	-------

22

OBSERVING PROCESS META-DATA

- Can inspect the number of **CONTEXT SWITCHES** made by a process
- Let's run mem.c (from chapter 2)
- `cat /proc/{process-id}/status`

```

Speculation_Store_Bypass:   thread vulnerable
Cpus_allowed:               ff
Cpus_allowed_list:          0-7
Mems_allowed:               00000000,00000001
Mems_allowed_list:          0
voluntary_ctxt_switches:    1372
nonvoluntary_ctxt_switches: 18
wllloyd@com.e7.2
```

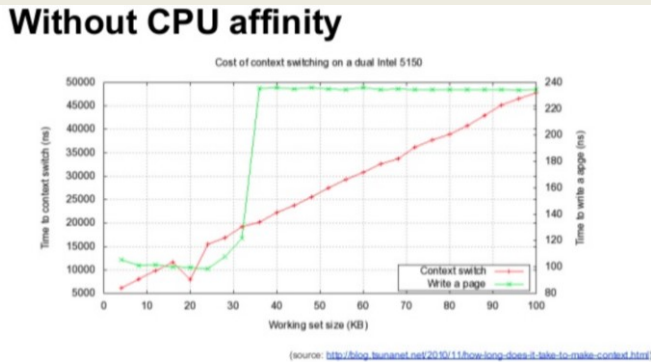
- `proc "status"` is a virtual file generated by Linux
- Provides a report with process related meta-data
- **What appears to happen to the number of context switches the longer a process runs? (mem.c)**

October 7, 2021	TCCS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.23
-----------------	---	-------

23

CONTEXT SWITCH

- **How long does a context switch take?**
- 10,000 to 50,000 ns (.01 to .05 ms)
- 2,000 context switches is near 100ms



October 7, 2021	TCCS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.24
-----------------	---	-------

24

W When a process is in this state, it is advantageous for the Operating System to perform a **CONTEXT SWITCH** to perform other work

RUNNING READY BLOCKED All of the above None of the above

October 7, 2021 TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma Total Re L3.25

25

QUESTION: WHEN TO CONTEXT SWITCH

- When a process is in this state, it is advantageous for the Operating System to perform a **CONTEXT SWITCH** to perform other work:
 - (a) **RUNNING**
 - (b) **READY**
 - (c) **BLOCKED**
 - (d) **All of the above**
 - (e) **None of the above**

October 7, 2021 TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma L3.26

26

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0
- Chapter 4: Processes
 - Process states, context switches
 - **Kernel data structures for processes and threads**
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021

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

L3.27

27

PROCESS DATA STRUCTURES

- OS provides data structures to track process information
 - **Process list**
 - Process Data
 - State of process: Ready, Blocked, Running
 - **Register context**
- **PCB (Process Control Block)**
 - A C-structure that contains information about each process

October 7, 2021

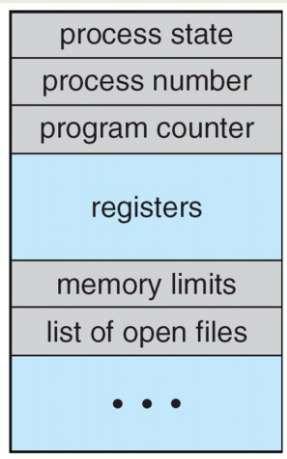
TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma

L3.28

28

STRUCT TASK_STRUCT PROCESS CONTROL BLOCK

- Process Control Block (PCB)
- Key data regarding a process



October 7, 2021

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

L3.29

29

XV6 KERNEL DATA STRUCTURES

- xv6: pedagogical implementation of Linux
- Simplified structures shown in book

```

// the registers xv6 will save and restore
// to stop and subsequently restart a process
struct context {
  int eip; // Index pointer register
  int esp; // Stack pointer register
  int ebx; // Called the base register
  int ecx; // Called the counter register
  int edx; // Called the data register
  int esi; // Source index register
  int edi; // Destination index register
  int ebp; // Stack base pointer register
};

// the different states a process can be in
enum proc_state { UNUSED, EMBRYO, SLEEPING,
  RUNNABLE, RUNNING, ZOMBIE };

```

October 7, 2021

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

L3.30

30

XV6 KERNEL DATA STRUCTURES - 2

```
// the information xv6 tracks about each process
// including its register context and state
struct proc {
    char *mem;           // Start of process memory
    uint sz;            // Size of process memory
    char *kstack;       // Bottom of kernel stack
                        // for this process

    enum proc_state state; // Process state
    int pid;             // Process ID
    struct proc *parent; // Parent process
    void *chan;         // If non-zero, sleeping on chan
    int killed;         // If non-zero, have been killed
    struct file *ofile[NOFILE]; // Open files
    struct inode *cwd;  // Current directory
    struct context context; // Switch here to run process
    struct trapframe *tf; // Trap frame for the
                        // current interrupt
};
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.31
-----------------	---	-------

31

LINUX: STRUCTURES

- **struct task_struct**, equivalent to **struct proc**
 - The Linux process data structure
 - Kernel data type (i.e. record) that describes individual Linux processes
 - Structure is **VERY LARGE: 10,000+ bytes**
 - Defined in:
/usr/src/linux-headers-{kernel version}/include/linux/sched.h
 - Ubuntu 20.04 w/ kernel version 5.11, **LOC: 657 – 1394**
 - Ubuntu 20.04 w/ kernel version 4.4, **LOC: 1391 – 1852**

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.32
-----------------	---	-------

32

STRUCT TASK_STRUCT

- Key elements (e.g. PCB) in Linux are captured in struct task_struct: (LOC from Linux kernel v 5.11)
- **Process ID**
- pid_t pid; LOC #857
- **Process State**
- /* -1 unrunnable, 0 runnable, >0 stopped: */
- volatile long state; LOC #666
- **Process time slice**
how long the process will run before context switching
- Struct sched_rt_entity used in task_struct contains timeslice:
 - struct sched_rt_entity rt; LOC #710
 - unsigned int time_slice; LOC #503

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.33
-----------------	---	-------

33

STRUCT TASK_STRUCT - 2

- **Address space of the process:**
- “mm” is short for “memory map”
- struct mm_struct *mm; LOC #779
- **Parent process**, that launched this one
- struct task_struct __rcu *parent; LOC #874
- **Child processes** (as a list)
- struct list_head children; LOC #879
- **Open files**
- struct files_struct *files; LOC #981

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.34
-----------------	---	-------

34

LINUX STRUCTURES - 2

- List of Linux data structures:
<http://www.tldp.org/LDP/tlk/ds/ds.html>

- Description of process data structures:
<https://learning.oreilly.com/library/view/linux-kernel-development/9780768696974/cover.html>
3rd edition is online (dated from 2010):
See chapter 3 on Process Management

Safari online – accessible using UW ID SSO login
Linux Kernel Development, 3rd edition
Robert Love
Addison-Wesley

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.35
-----------------	---	-------

35

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0

- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- **Chapter 5: Process API**
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.36
-----------------	---	-------

36


**WE WILL RETURN AT
2:50PM**



October 7, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L3.37

37

**CHAPTER 5:
C PROCESS API**



October 7, 2021 TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma L3.38

38


OBJECTIVES – 10/7

- Questions from 10/5
 - C Review Survey – Due Sunday Oct 10
 - Student Background Survey
 - Virtual Machine Survey: VM requests sent to S. Rondeau
 - Assignment 0

 - Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
 - Chapter 5: Process API
 - **fork()** wait(), exec()
- | | | |
|-----------------|---|-------|
| October 7, 2021 | TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L3.39 |
|-----------------|---|-------|

39

fork()

- Creates a new process - think of “a fork in the road”
 - “Parent” process is the original
 - Creates “child” process of the program from the **current execution point**
 - Book says “pretty odd”
 - Creates a **duplicate** program instance (these are **processes!**)
 - **Copy** of
 - Address space (memory)
 - Register
 - Program Counter (PC)
 - Fork returns
 - child PID to parent
 - 0 to child
- 
- | | | |
|-----------------|---|-------|
| October 7, 2021 | TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L3.40 |
|-----------------|---|-------|

40

FORK EXAMPLE

▪ p1.c

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

int main(int argc, char *argv[]){
    printf("hello world (pid:%d)\n", (int) getpid());
    int rc = fork();
    if (rc < 0) { // fork failed; exit
        fprintf(stderr, "fork failed\n");
        exit(1);
    } else if (rc == 0) { // child (new process)
        printf("hello, I am child (pid:%d)\n", (int) getpid());
    } else { // parent goes down this path (main)
        printf("hello, I am parent of %d (pid:%d)\n",
            rc, (int) getpid());
    }
    return 0;
}
```



41

FORK EXAMPLE - 2

▪ Non deterministic ordering of execution

```
prompt> ./p1
hello world (pid:29146)
hello, I am parent of 29147 (pid:29146)
hello, I am child (pid:29147)
prompt>
```

or

```
prompt> ./p1
hello world (pid:29146)
hello, I am child (pid:29147)
hello, I am parent of 29147 (pid:29146)
prompt>
```

▪ CPU scheduler determines which to run first

42

:(){ :|: & }::

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.43
-----------------	---	-------

43

OBJECTIVES – 10/7

- Questions from 10/5
- C Review Survey – Due Sunday Oct 10
- Student Background Survey
- Virtual Machine Survey: VM requests sent to S. Rondeau
- Assignment 0

- Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
- Chapter 5: Process API
 - fork(), wait(), exec()

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.44
-----------------	---	-------

44

wait()

- wait(), waitpid()
- Called by parent process
- Waits for a child process to finish executing
- Not a sleep() function
- Provides some ordering to multi-process execution



October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.45
-----------------	---	-------

45

FORK WITH WAIT

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[]){
    printf("hello world (pid:%d)\n", (int) getpid());
    int rc = fork();
    if (rc < 0) { // fork failed; exit
        fprintf(stderr, "fork failed\n");
        exit(1);
    } else if (rc == 0) { // child (new process)
        printf("hello, I am child (pid:%d)\n", (int) getpid());
    } else { // parent goes down this path (main)
        int wc = wait(NULL);
        printf("hello, I am parent of %d (wc:%d) (pid:%d)\n",
            rc, wc, (int) getpid());
    }
    return 0;
}
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.46
-----------------	---	-------

46

FORK WITH WAIT - 2

- **Deterministic ordering of execution**

```
prompt> ./p2
hello world (pid:29266)
hello, I am child (pid:29267)
hello, I am parent of 29267 (wc:29267) (pid:29266)
prompt>
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.47
-----------------	---	-------

47

FORK EXAMPLE

- **Linux example**

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.48
-----------------	---	-------

48

OBJECTIVES – 10/7

- Questions from 10/5
 - C Review Survey – Due Sunday Oct 10
 - Student Background Survey
 - Virtual Machine Survey: VM requests sent to S. Rondeau
 - Assignment 0

 - Chapter 4: Processes
 - Process states, context switches
 - Kernel data structures for processes and threads
 - Chapter 5: Process API
 - fork(), wait(), **exec()**
- | | | |
|-----------------|---|-------|
| October 7, 2021 | TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L3.49 |
|-----------------|---|-------|

49

exec()

- Supports running an external program **by “transferring control”**
 - 6 types: execl(), execlp(), execl(), execv(), execvp(), execvpe()

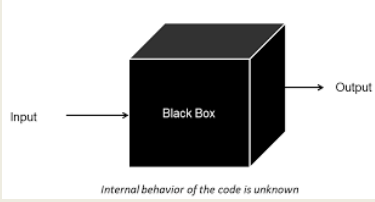
 - execl(), execlp(), execl(): const char *arg **(example: execl.c)**
Provide cmd and args as individual params to the function
Each arg is a pointer to a null-terminated string
ODD: pass a variable number of args: (arg0, arg1, .. argn)

 - Execv(), execvp(), execvpe() **(example: exec.c)**
Provide cmd and args as an Array of pointers to strings
Strings are null-terminated
First argument is name of command being executed
Fixed number of args passed in
- | | | |
|-----------------|---|-------|
| October 7, 2021 | TCSS422: Operating Systems [Fall 2021]
School of Engineering and Technology, University of Washington - Tacoma | L3.50 |
|-----------------|---|-------|

50

EXEC() - 2

- Common use case:
 - Write a new program which wraps a legacy one
 - Provide a new interface to an old system: Web services
 - Legacy program thought of as a “black box”
- We don't want to know what is inside... 😊



October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.51
-----------------	---	-------

51

EXEC EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

int main(int argc, char *argv[]){
    printf("hello world (pid:%d)\n", (int) getpid());
    int rc = fork();
    if (rc < 0) { // fork failed; exit
        fprintf(stderr, "fork failed\n");
        exit(1);
    } else if (rc == 0) { // child (new process)
        printf("hello, I am child (pid:%d)\n", (int) getpid());
        char *myargs[3];
        myargs[0] = strdup("wc"); // program: "wc" (word count)
        myargs[1] = strdup("p3.c"); // argument: file to count
        myargs[2] = NULL; // marks end of array
        ...
    }
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.52
-----------------	---	-------

52

EXEC EXAMPLE - 2

```
...  
➔ execvp(myargs[0], myargs); // runs word count  
printf("this shouldn't print out");  
} else { // parent goes down this path (main)  
int wc = wait(NULL);  
printf("hello, I am parent of %d (wc:%d) (pid:%d)\n",  
rc, wc, (int) getpid());  
}  
return 0;  
}
```

```
prompt> ./p3  
hello world (pid:29383)  
hello, I am child (pid:29384)  
29 107 1030 p3.c  
hello, I am parent of 29384 (wc:29384) (pid:29383)  
prompt>
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.53
-----------------	---	-------

53

EXEC WITH FILE REDIRECTION (OUTPUT)

```
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <string.h>  
#include <fcntl.h>  
#include <sys/wait.h>  
  
int  
main(int argc, char *argv[]){  
int rc = fork();  
if (rc < 0) { // fork failed; exit  
fprintf(stderr, "fork failed\n");  
exit(1);  
} else if (rc == 0) { // child: redirect standard output to a file  
➔ close(STDOUT_FILENO);  
open("./p4.output", O_CREAT|O_WRONLY|O_TRUNC, S_IRWXU);  
...  
}}
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.54
-----------------	---	-------

54

FILE MODE BITS



```
S_IRWXU
read, write, execute/search by owner
S_IRUSR
read permission, owner
S_IWUSR
write permission, owner
S_IXUSR
execute/search permission, owner
S_IRWXG
read, write, execute/search by group
S_IRGRP
read permission, group
S_IWGRP
write permission, group
S_IXGRP
execute/search permission, group
S_IRWXO
read, write, execute/search by others
S_IROTH
read permission, others
S_IWOTH
write permission, others
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.55
-----------------	---	-------

55

EXEC W/ FILE REDIRECTION (OUTPUT) - 2

```
...
// now exec "wc"...
char *myargs[3];
myargs[0] = strdup("wc"); // program: "wc" (word count)
myargs[1] = strdup("p4.c"); // argument: file to count
myargs[2] = NULL; // marks end of array
execvp(myargs[0], myargs); // runs word count
} else { // parent goes down this path (main)
    int wc = wait(NULL);
}
return 0;
}
```



```
prompt> ./p4
prompt> cat p4.output
32 109 846 p4.c
prompt>
```

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.56
-----------------	---	-------

56

W Which Process API call is used to launch a different program from the current program?

Fork() Exec() Wait() None of the above All of the above

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

57

QUESTION: PROCESS API

- Which Process API call is used to launch a different program from the current program?
- (a) Fork()
- (b) Exec()
- (c) Wait()
- (d) None of the above
- (e) All of the above

October 7, 2021	TCSS422: Operating Systems [Fall 2021] School of Engineering and Technology, University of Washington - Tacoma	L3.58
-----------------	---	-------

58

QUESTIONS



59