

REVIEW: SCHEDULING AND MEMORY

CS 537, Spring 2025

Discussion Section

VIRTUAL MEMORY REVIEW

- Goals for Memory Virtualization

VIRTUAL MEMORY REVIEW

- Strategies for memory virtualization
 - Static relocation
 - Base
 - Base + bounds
 - Segments
 - Paging

MAKING PAGING EFFICIENT

- Reducing the number of page faults
- Making page tables smaller

ADDRESS TRANSLATION EXAMPLE

VA 0: 0x6baa (decimal: 27562) --> VALID: 0x1e fbc (decimal: 126908)

VA 1: 0x4248 (decimal: 16968) --> VALID: 0x1c65a (decimal: 116314)

VA 2: 0x82e2 (decimal: 33506) --> SEGMENTATION VIOLATION

VA 3: 0x67a9 (decimal: 26537) --> VALID: 0x1ebbb (decimal: 125883)

VA 4: 0xc8a7 (decimal: 51367) --> SEGMENTATION VIOLATION

VA 5: 0x2568 (decimal: 9576) --> VALID: 0x2568 (decimal: 9576)

- Base register?
- What is physical address for VA 0x4826

PAGE TABLE EXAMPLE

- How big are page tables?
 - 32-bit addresses, 4kB pages, 4-byte PTEs
 - 32-bit addresses, 4KB pages, 8-byte PTEs
 - 20-bit addresses, 512-byte pages, 4-byte PTEs
 - 12-bit addresses, Number of Virtual Pages: 128, Page Size: 1KB

TLB EXAMPLE

Assume 16-bit virtual addresses, 512 pages in virtual address space. Ignore instruction references

- 1-entry TLB: sequential access to array of contiguous 4-byte integers. What is TLB miss rate?
- 4-entry TLB, same workload
- 1-entry TLB, access every 4th element of array
- How big should TLB be to access 128 element array repeatedly with 100% hit rate?

PAGE REPLACEMENT

- Policies:
 - LRU, FIFO, Random, Clock, Belady's anomaly

QUESTIONS?

SUMMARY: VIRTUAL MEMORY

Abstraction: Virtual address space with code, heap, stack

Address translation

- Contiguous memory: base, bounds, segmentation
- Using fixed sizes pages with page tables

Challenges with paging

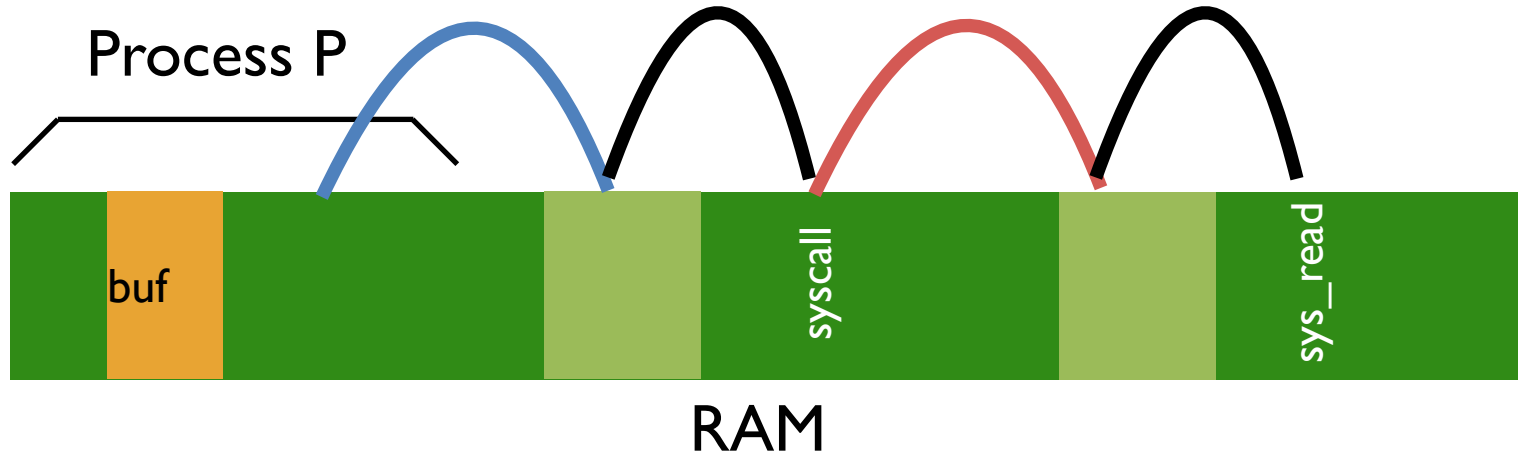
- Extra memory references: avoid with TLB
- Page table size: avoid with multi-level paging, inverted page tables etc.

Larger address spaces: Swapping mechanisms, policies (LRU, Clock)

SYSTEM CALLS AND PROCESSES

- Processes vs Programs
 - What is added?
- Process states: ready, running, blocked
- Privileged operations & limited direct execution
- Fork/exec
- System calls
 - What are the steps
- Multi-tasking
 - Cooperative
 - Preemptive with timer interrupts

SYSTEM CALL



Kernel can access user memory to fill in user buffer
return-from-trap at end to return to Process P

CONTEXT SWITCH: WHAT MUST BE SAVED?

Dispatcher must track context of process when not running

- Save context in **process control block (PCB)** (or, process descriptor)

What information is stored in PCB?

- PID
- Process state (i.e., running, ready, or blocked)
- Execution state (all registers, PC, stack ptr)
- Scheduling priority
- Accounting information (parent and child processes)
- Credentials (which resources can be accessed, owner)
- Pointers to other allocated resources (e.g., open files)

Requires special hardware support

- Hardware saves process PC and PSR on interrupts

UNIX PROCESS CREATION

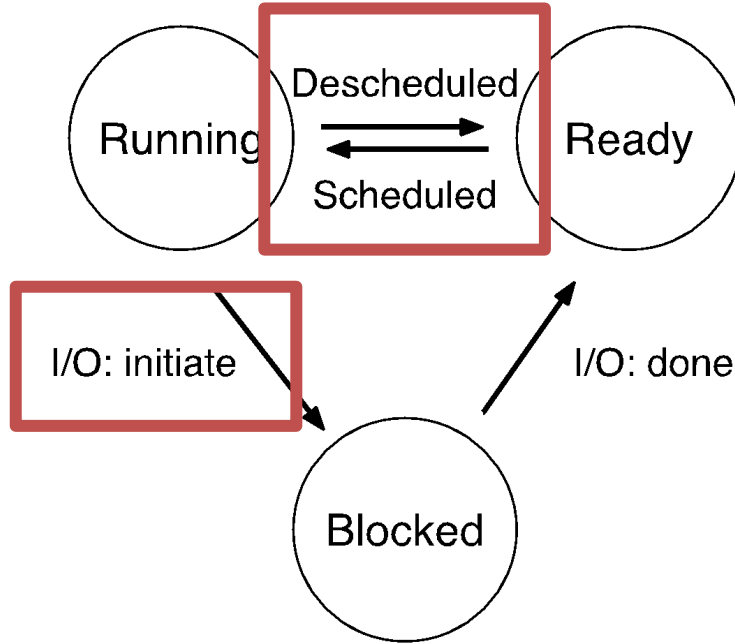
How are Unix shells implemented?

```
While (1) {
    Char *cmd = getcmd();
    Int retval = fork();
    If (retval == 0) {
        // This is the child process
        // Setup the child's process environment here
        // E.g., where is standard I/O, how to handle signals?
        exec(cmd);
        // exec does not return if it succeeds
        printf("ERROR: Could not execute %s\n", cmd);
        exit(1);
    } else {
        // This is the parent process; Wait for child to finish
        int pid = retval;
        wait(pid);
    }
}
```

FORK EXAMPLE: WHAT WILL BE PRINTED?

Assume: parent PID = 1, PIDs
allocated sequentially

```
int child1_pid = fork();  
int child2_pid = fork();  
printf("c1 = %d, c2 = %d\n",  
       child1_pid, child2_pid  
);
```



POLICY ?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8		
B	2	4		
C	5	7		

FIFO scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8	0	8-0
B	2	4	8-2	8-2+4
C	5	7	8-2+4+2-5	8+4-5+7

FIFO scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8	0	8
B	2	4	6	10
C	5	7	7	14

FIFO scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8		
B	2	4		
C	5	7		

SJF scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8	0	8
B	2	4	6	10
C	5	7	7	14

SJF scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8		
B	2	4		
C	5	7		

STCF scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8	0	12
B	2	4	0	4
C	5	7	7	14

STCF scheduling?

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8		
B	2	4		
C	5	7		

RR Timeslice 1

SCHEDULING EXAMPLE

Job Name	Arrival Time	CPU burst (seconds)	Response Time	Turnaround Time
A	0	8	0	17
B	2	4	0	9
C	5	7	0	14

RR Timeslice 1

AABABCABCACACACC

MULTILEVEL FEEDBACK QUEUE

