JEOPARDY!

CIS 5480 Midterm Review

February 27, 2025

JEOPARDY BOARD

FINAL JEOPARDY

Process Management	Signals	File Descriptors & I/O	File Systems	Concurrency & Synchronization	CPU Scheduling
$100	$100	$100	$100	$100	$100
$200	$200	$200	$200	$200	$200
$300	~~$300~~	$300	$300	$300	$300
$400	$400	$400	$400	$400	$400
$500	$500	$500	$500	$500	$500

Process Management - $100 Question

What does the wait() system call do in a parent process? Why is calling wait() important after forking a child process?

Click to see answer 

Process Management - $100 Answer

wait() makes the parent process pause execution until one of its child processes terminates. It allows the parent to retrieve the child's exit status and prevents the creation of zombie processes.

Click to return to Jeopardy Board

Process Management - $200 Question

What are the usual return values of fork() in the parent and child processes upon execution?

Click to see answer 

Process Management - $200 Answer

The child process returns 0.
The parent process returns
the pid of the child.
On failure, it returns -1.

Click to return to Jeopardy Board

Process Management - $300 Question

How does the exec() family of functions differ from fork(), and why are they often used together?

Click to see answer

Process Management - $300 Answer

fork() creates a new process by duplicating the current one.

exec() replaces the current process with a new program.

They are commonly used together so that the child process created by fork() can immediately load and run a different program using exec().

Click to return to Jeopardy Board

Process Management - $400 Question

A simple shell spawns a child process to run a foreground job using fork(). However, the user notices that pressing Ctrl+C does not terminate the child process but instead affects the shell itself. What mistake in terminal control is likely causing this issue, and how can it be fixed?

Click to see answer

Process Management - $400 Answer

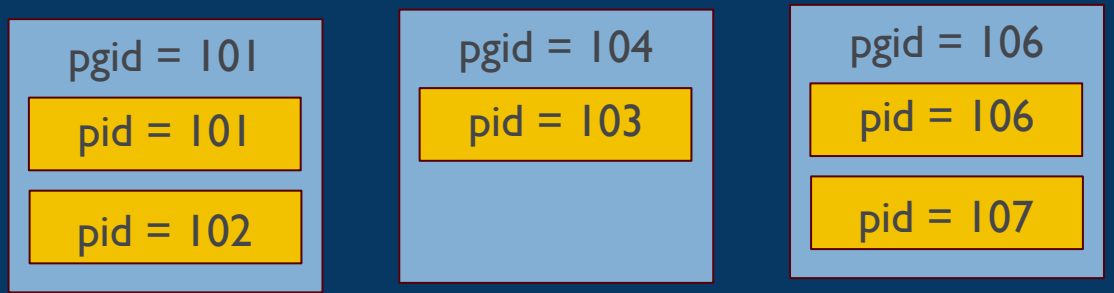
The shell likely failed to call tcsetpgrp() to assign the child process's group as the terminal's foreground process group.
To fix the issue, we would need to run:
tcsetpgrp(STDIN_FILENO, child_pgid);
After forking.

Click to return to Jeopardy Board

Process Management - $500 Question

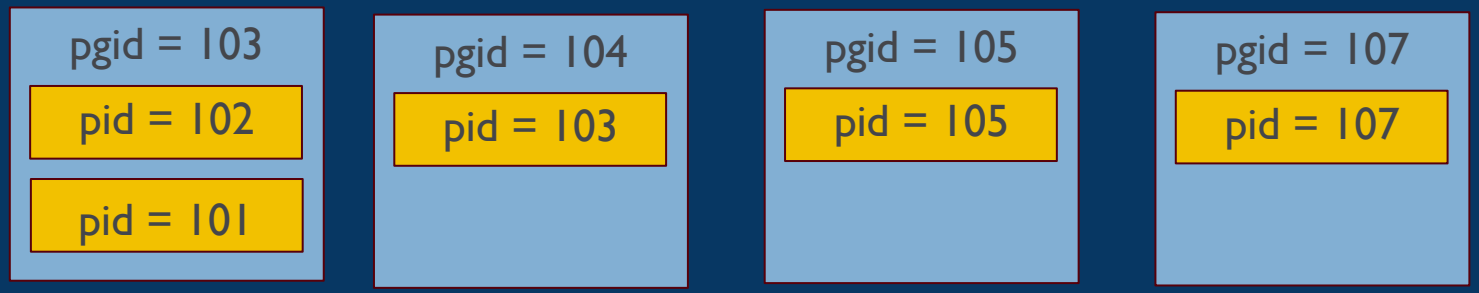
What are the resulting process groups and process assignments after executing the following code? Assume that the operating system assigns process IDs (PIDs) based on the first available integer greater than 100.

```c
int main() {
    pid_t pid = getpid();
    if (pid == 103) {
        pid_t child = fork();
        if (child == 0) {
            setpgid(0, 0);
        }
    }
    kill(101, SIGTERM);
    if (pid == 102) {
        setpgid(0, 103);
        pid_t child = fork();
        if (child != 0) {
            setpgid(107, 0);
        }
    }
    kill(-106, SIGTERM);
    return 0;
}
```



Click to see answer

Process Management - $500 Answer



Click to return to Jeopardy Board

Signals - $100 Question

What does the alarm() system call do, and how can it be used to implement a timeout mechanism?

Click to see answer

Signals - $100 Answer

Alarm schedules a SIGALRM to be delivered to the process after a given number of seconds. To implement a timeout, we call alarm(s)

Click to return to Jeopardy Board

Signals - $200 Question

Describe the difference between signals and interrupts; say a user enters CTRL+C. How is SIGINT actually sent/handled?

Click to see answer 

Signals - $200 Answer

Interrupts originate from hardware, whereas signals are higher-level (software). SIGINT is sent by the OS, which checks the foreground process' PCB for the signal disposition (term, ign, …) or a handler if one is set.

Click to return to Jeopardy Board

Signals - $300 Question

Type question here

Click to see answer 

Signals - $300 Answer

Type answer here

Click to return to Jeopardy Board 

Signals - $400 Question

A process sets a signal handler for SIGUSR with sigaction(). Inside the handler, a global variable is modified. What are some potential issues if the handler is preempted by another signal

Click to see answer

Signals - $400 Answer

Race condition: concurrent access to the global variable, inconsistent state of the global var

Click to return to Jeopardy Board

Signals - $500 Question

```
5   void handler(int sig) {
6       printf("Caught SIGINT!\n");
7   }
8
9   int main() {
10      sigset_t block_set, old_set;
11
12      signal(SIGINT, handler);
13
14      sigemptyset(&block_set);
15      sigaddset(&block_set, SIGINT);
16      sigprocmask(SIG_BLOCK, &block_set, &old_set);
17
18      sleep(3);
19
20      sigprocmask(SIG_SETMASK, &old_set, NULL);
21
22      while (1) sleep(1);
23
24      return 0;
25  }
26
```

Explain what this example does; if CTRL+C is pressed once every second for 6 seconds, how many times is “Caught SIGINT!” printed?

Click to see answer

Signals - $500 Answer

Four times.
While SIGINT is blocked, the OS does not queue standard signals. So only one of the three SIGINTs during the first three seconds will be handled. In the remaining three seconds, the SIGINTs will be handled correctly.

Click to return to Jeopardy Board

File Descriptors & I/O - $100 Question

How does the file descriptor table function in a Unix-like operating system? What is stored in it?

Click to see answer 

File Descriptors & I/O - $100 Answer

Each process maintains its own file descriptor table, which maps these integers to the underlying open files or devices, including associated metadata like file offsets and access modes.

Click to return to Jeopardy Board

File Descriptors & I/O - $200 Question

What roles do the open() and close() system calls play in managing file descriptors?

Click to see answer 

File Descriptors & I/O - $200 Answer

The open() system call opens a file or device, returning a file descriptor that references it, while close() terminates the association with that file descriptor, freeing the resource for future use.

Click to return to Jeopardy Board

File Descriptors & I/O - $300 Question

How does file descriptor inheritance work when a process calls fork(), and what impact does it have on I/O redirection?

Click to see answer

File Descriptors & I/O - $300 Answer

When a process forks, the child process inherits a copy of the parent's file descriptor table. This inheritance means both processes can access the same open files, which facilitates I/O redirection and inter-process communication.

Click to return to Jeopardy Board

File Descriptors & I/O - $400 Question

What is the difference between using dup() and dup2() for file descriptor duplication, and how are they typically used in shell redirection?

Click to see answer

File Descriptors & I/O - $400 Answer

dup(fd) returns the lowest available new file descriptor referencing the same resource, while dup2(oldfd, newfd) forces duplication into a specific file descriptor number (closing it first if open). Shells use dup2() to redirect standard I/O streams to files or pipes.

Click to return to Jeopardy Board

File Descriptors & I/O - $500 Question

How do Unix-like shells implement a multi-stage pipeline (cmd1 | cmd2 | cmd3) using file descriptors, and what role do the pipe ends play?

Click to see answer

File Descriptors & I/O - $500 Answer

For each pipeline stage, the shell uses pipe() to create a pipe with read and write file descriptors. It then forks each command, connecting the write end of one pipe to the read end of the next using dup2().

Click to return to Jeopardy Board

File Systems - $100 Question

What is the smallest unit of work for a file system (R/W)?

a. A bit
b. A byte
c. A block
d. A page

Click to see answer

File Systems - $100 Answer

C. A block

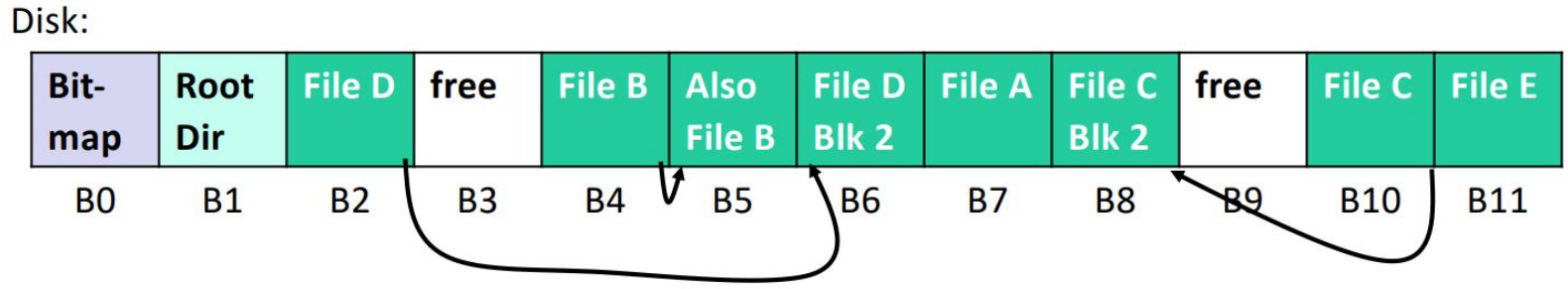
Even if you want to change just one byte in a file, you must write an entire block of that file.

Click to return to Jeopardy Board

File Systems - $200 Question

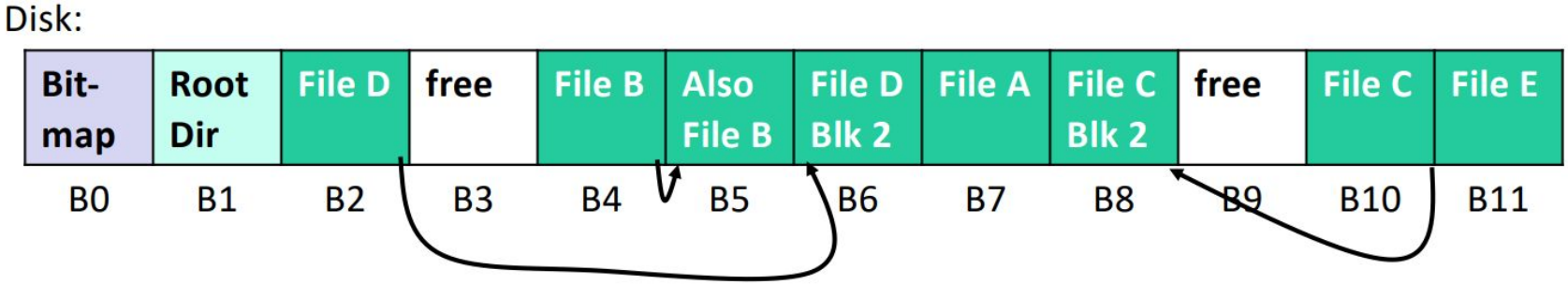
In an implicit linked list architecture of the file system…

What if I want to grow File D by 2 blocks?

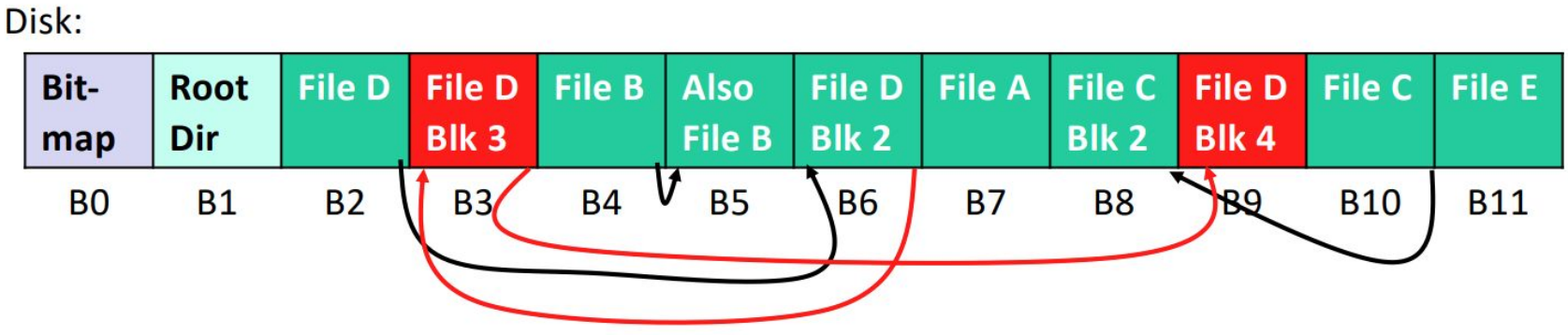


Click to see answer

File Systems - $200 Answer



- Scan the bitmap to find which blocks are free
- Allocate the blocks and set up pointers to them



Click to return to Jeopardy Board

File Systems - $300 Question

Block #	File X Block Y
0	FAT
1	Root directory
2	File A Block 1
3	?
4	File B Block 1
5	Empty
6	?
7	File C Block 1
8	?
9	?
10	File D Block 1

Block #	Next
0	BITMAP/SPECIAL
1	END
2	6
3	9
4	END
5	EMPTY / UNUSED
6	3
7	END
8	END
9	END
10	8

Click to see answer

File Systems - $300 Answer

Block #	File X Block Y
0	FAT
1	Root directory
2	File A Block 1
3	*File A Block 3*
4	File B Block 1
5	Empty
6	*File A Block 2*
7	File C Block 1
8	*File D Block 2*
9	*File A Block 4*
10	File D Block 1

Block #	Next
0	BITMAP/SPECIAL
1	END
2	6
3	9
4	END
5	EMPTY / UNUSED
6	3
7	END
8	END
9	END
10	8

Click to return to Jeopardy Board

File Systems - $400 Question

With an Inode struct like below, what's the largest file size possible if...

- Each block is 512 bytes
- Each block_no_t is 4 bytes?

(Express your answer as an equation)

```
struct inode_st {
  attributes_t metadata;
  block_no_t blocks[12];
  block_no_t *single_ind;
  block_no_t **double_ind;
  block_no_t ***triple_ind;
};
```

Click to see answer

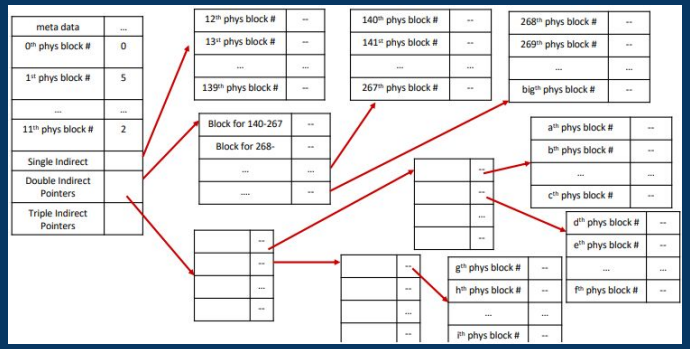
File Systems - $400 Answer

```
struct inode_st {
  attributes_t metadata;
  block_no_t blocks[12];
  block_no_t *single_ind;
  block_no_t **double_ind;
  block_no_t ***triple_ind;
};
```

- Each block is 512 bytes
- Each block_no_t is 4 bytes

Direct blocks: 12 * 512 = 6,144 bytes
Single indirect: 512/4 = 128 pointers, 128 * 512 = 65,536 bytes
Double indirect: 128 * 128 * 512 = 8,388,608 bytes
Triple indirect: 128^3 * 512 = 1,073,741,824 bytes

Total: 6144 + 65536 + 8388608 + 1073741824 ≈ 1.01GB



Click to return to Jeopardy Board

File Systems - $500 Question

Assume…

- A block is 1,024 bytes
- Each inode is 128 bytes
- There are 12 directory entries (dirents) in each inode

```
struct dirent {
    ino_t d_ino;       /* File inode number */
    char d_name[];     /* Null terminated name of file */
    uint8_t d_type;    /* Indicator of file type ONLY IN ext2, 3, 4 */
}
```

maximum

What is the ~~minimum~~ number of directory entries (struct dirents) that must be traversed to resolve the path `/dir_one/dir_two/dir_three/file.txt` and access the data blocks of file.txt in the ~~best-case~~ scenario? Assume you have not accessed the root inode.

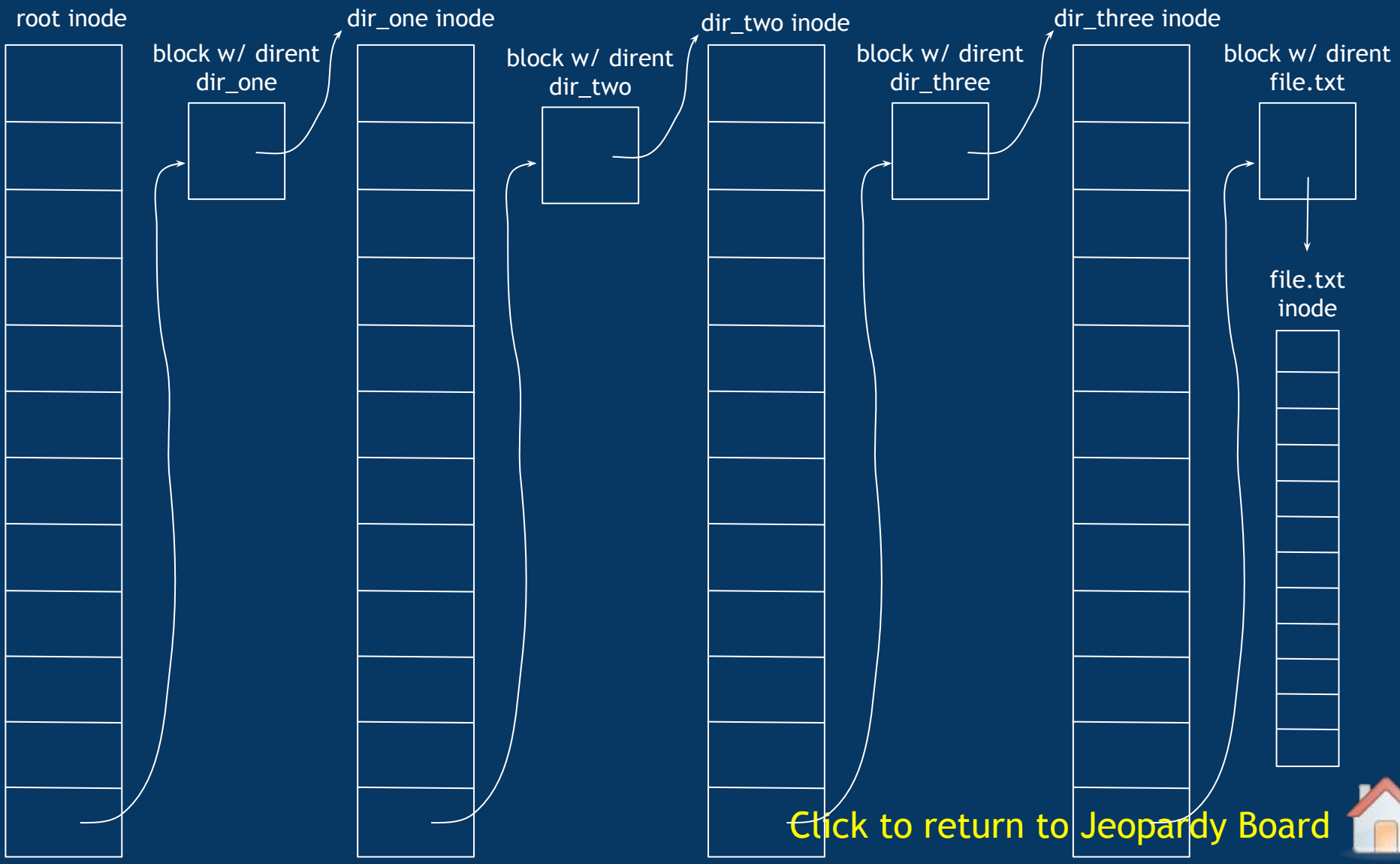
worst-case 😈

(assume file.txt does exist in this path)

Click to see answer

File Systems - $500 Answer

5 inodes
48 direnTs



Click to return to Jeopardy Board

Concurrency & Synchronization - $100 Question

What is the unit of scheduling in most modern OS?

a. A process

b. A thread

c. A Procedure

Click to see answer 

Concurrency & Synchronization - $100 Answer

b. A thread

Click to return to Jeopardy Board

Concurrency & Synchronization - $200 Question

With a 4-core machine, what's the most amount of performance boost I can get when executing a task with multiple threads? Using *parallelism* or *concurrency*?

Click to see answer 

Concurrency & Synchronization - $200 Answer

4x with parallelism

Physical parallelism is limited by the number of available cores. With 4 cores, only 4 threads can truly execute instructions simultaneously.

Click to return to Jeopardy Board

Concurrency & Synchronization - $300 Question

Which of the following are shared
(1) between processes
(2) between threads but not processes
(3) by neither of the two?

a. Stack
b. Heap
c. Registers
d. Stack pointer
e. Program counter
f. Pipes via pipe()

Click to see answer 

Concurrency & Synchronization - $300 Answer

a. Stack
b. Heap
c. Registers
d. Stack pointer
e. Program counter
f. Pipes via pipe()

(1). Between processes: f
(2) Between threads but not processes: b
(3) Neither: acde

Click to return to Jeopardy Board

Concurrency & Synchronization - $400 Question

Describe a scenario where using processes would be preferable to threads despite their higher overhead.

Click to see answer

Concurrency & Synchronization - $400 Answer

One advantage for using processes is the isolation: if one process crashes, others can keep working.

Scenario: web server handling multiple client requests with separate processes.

Click to return to Jeopardy Board

Concurrency & Synchronization - $500 Question

In a multithreaded program where one thread attempts to increment a global counter variable 1000 times and another thread attempts to decrement it 1000 times.

Why might the final value not be 0?

Click to see answer

Concurrency & Synchronization - $500 Answer

Race condition!

An example scenario:

- Thread A reads the counter value as 10
- Thread B reads the counter value as 10
- Thread A adds 1 and writes back 11
- Thread B subtracts 1 and writes back 9

Click to return to Jeopardy Board

Scheduling - $100 Question

Which scheduling algorithm(s) runs the risk of "starving" its jobs?

Click to see answer

Scheduling - $100 Answer

Shortest Job First (SJF)
Priority Round Robin

Click to return to Jeopardy Board

Scheduling - $200 Question

If you're guaranteed that all incoming jobs take approximately the same amount of time to complete, which scheduling algorithm makes the most sense to implement and why?

Click to see answer

Scheduling - $200 Answer

First Come First Served (FCFS):

- Easiest to implement
- Consistent duration of jobs negates any positive benefits of other scheduling algorithms.
- Any preemptive scheduling algorithms will cost more time with the extra context-switching

Click to return to Jeopardy Board

Scheduling - $300 Question

Given the following jobs and their estimated lengths, what time quantum would you choose for a Round Robin Scheduler?

Job A: 2-10ns
Job B: 4-6ns
Job C: 1-17ns
Job D: 6-7ns
Job E: 8-13ns
Job F: 1-2ns
Job G: 3-14ns
Job H: 15-18ns
Job I: 5-11ns
Job J: 6-9ns

Click to see answer

Scheduling - $300 Answer

If you want 80-90% of jobs to finish within 1 attempt: need to use upper estimate

80% of jobs finish within 14ns
90% of jobs finish within 17ns
So 14ns <= TQ <= 17n

Click to return to Jeopardy Board

Scheduling - $400 Question

Rank the following in order of preference in priority round-robin and explain your reasoning:

1. Involving little to no I/O
2. Involving a lot of I/O, but split into smaller chunks
3. One long sequence of I/O accesses at the first half of the job, but no I/O in the last half

Ties are allowed!

Click to see answer 

Scheduling - $400 Answer

Small I/O bursts >= One long sequence of I/O > No I/O

Want to make most utilization of I/O devices when needed - scheduler can context-switch to another thread that needs CPU while I/O is accessed, and switch back quickly if I/O access is relatively short. If no I/O bursts are needed, then nothing useful is happening while the thread is in queue!

Click to return to Jeopardy Board

Scheduling - $500 Question

Job Name	Arrival Time	Running Time
Swellow	2	11
Pidove	0	8
Fletchling	12	20
Starly	7	15
Doduo	10	4

Using a Round Robin scheduling algorithm and a time quantum of 8…

1. what is the finishing time for each job?
2. What is the average waiting time?
3. Is 8 the best time quantum to use given the current info?

Click to see answer 

Scheduling - $500 Answer

Job Name	Finishing Time	Wait Time
Swellow	39	39 - 2 - 11 = 26
Pidove	8	8 - 0 - 8 = 0
Fletchling	58	58 - 12 - 20 = 26
Starly	46	46 - 7 - 15 = 24
Doduo	28	28 - 10 - 4 = 14

Average wait time: 18

No, this time quantum is quite short - only 40% of known jobs can finish within 1 attempt. A better time quantum would be between 15 and 19, inclusive.

Click to return to Jeopardy Board

Other questions you can add about scheduling:

Q: what scheduling algorithm can be implemented using an array of linked lists?
A: priority round robin (~200 pts)

FINAL

JEOPARDY!

Topic: Type topic here

Click to see question 

Final Jeopardy Question

Type question here

Click to see answer 

Final Jeopardy Answer

Type answer here

Click to return to Jeopardy Board