

! This quiz has been regraded; your score was not affected.

Midterm Quiz 1

Due Jan 30 at 11:59pm

Points 34

Questions 21

Available Jan 9 at 8am - Jan 30 at 11:59pm

Time Limit 120 Minutes

Allowed Attempts 3

Instructions

Instructions / Topic



This midterm quiz for CS 344 is 120 minutes long and covers materials from Jan 9th to Jan 25th (see [the schedule](https://secure-ai.systems/courses/OS1/W23/syllabus.html) [↗](https://secure-ai.systems/courses/OS1/W23/syllabus.html) (<https://secure-ai.systems/courses/OS1/W23/syllabus.html>)).

This quiz is unproctored. You are allowed to use whatever notes, lectures, websites, and books you have. You also have three chances to take this quiz and the highest score will be recorded. But you must observe the following rule: **do not take this test with any other students of this class, whether physically or virtually together.** Canvas makes it easy to see correlations between IP addresses, times, grades, and answer choices, so don't cheat - you'll get caught and have to face the penalties!

You are not allowed to receive external "human help" while taking this quiz. That also means that you are prohibited from posting questions on Stack Overflow or similar websites.

Please note that after you complete the quiz, Canvas will show you the points you scored, but *it won't show you the questions you got right or wrong*. This is a security feature to help prevent sharing answers, as everyone will be taking this class at different times.

This quiz was locked Jan 30 at 11:59pm.

Attempt History

	Attempt	Time	Score	Regraded
KEPT	Attempt 3	28 minutes	30 out of 34	30 out of 34
LATEST	Attempt 3	28 minutes	30 out of 34	30 out of 34
	Attempt 2	37 minutes	28.2 out of 34	29.2 out of 34
	Attempt 1	115 minutes	26.6 out of 34	26.6 out of 34

❗ Correct answers are hidden.

Score for this attempt: **30** out of 34

Submitted Jan 30 at 6:36pm

This attempt took 28 minutes.

Question 1

3 / 3 pts

Consider the following code snippet.

```
static int bank_balance = 1000;

int deposit(int amount, int account_balance)
{
    bank_balance += amount;
    return (account_balance + amount);
}

int main()
{
    int tom_balance = 100;
    int bob_balance = 200;

    // receive paychecks
    tom_balance = deposit(500, tom_balance);
    bob_balance = deposit(500, bob_balance);
    return 0;
}
```

What are the final balances of Tom / Bob / Bank?

☐ 600 / 700 / 1000

☐ 100 / 200 / 1000

☒ 600 / 700 / 2000

☐ 600 / 700 / 1500

☐ 100 / 700 / 1000

Question 2**3 / 3 pts**

Consider the following code snippet

```
int main()
{
    char *cptr = "Hello-world!";
    int *iptr = (int *) cptr;
    int cnt = 0;

    printf("%s\n", cptr);
    printf("%s\n", iptr);

    printf("%s\n", cptr+1);
    printf("%s\n", iptr+1);

    for (cnt = 0; cnt < 13; ++cnt)
    {
        printf("[loop] %s\n", cptr+cnt);
        printf("[loop] %s\n", iptr+cnt);
    }

    return 0;
}
```

Which of the following statements is false?

☐ The first and the second printf statements print "Hello-world!"

☒ In the for-loop, the first printf ("[loop] ...") reads data outside the "*cptr" string.

☐ The fourth printf statement prints "o-world!"

☐ The third printf statement prints "ello-world!"

☐ In the for-loop, the second printf ("[loop] ...") reads data outside the "*cptr" string.

Question 3**3 / 3 pts**

Consider the following code snippet.

```
int main()
{
    int bufsize = 10;
    char *buf = (char *) malloc(bufsize * sizeof(char));
    char *str = "Hello world!";

    strcpy(buf, str);

    printf("buf contains: %s [size: %d]\n", buf, (int) strlen(buf));
    printf("str contains: %s [size: %d]\n", str, (int) strlen(str));
    return 0;
}
```

Which of the following statements is true?

☐

The “*buf” is initialized to point to the memory space whose size is 11 bytes

☐

The second printf statement prints “str contains: Hello world! [size: 10]”

☒

“strcpy” function does not check the size of “*buf” and copies the 13 bytes from “str”

☐

The first printf statement prints “buf contains: Hello world! [size: 13]”

☐

The program frees the memory space “*buf” holds upon termination

Question 4

1 / 1 pts

Operating system is a software that lies between user application and hardware

☒

True

☐

False

Incorrect

Question 5 Original Score: 1 / 1 pts **Regraded Score: 1 / 1 pts****! This question has been regraded.**

Studying operating systems provides us with an understanding of how computers think

☐ True☒ False**Question 6****2 / 2 pts**

Which of the following statements is false?

☐

In 50s - 60s, computers are expensive and humans are cheap; thus, human operators work as operating systems to increase the job-to-job transition (e.g., a job = a process)

☐

In the 90s, users demanded complex functionalities, so the OS developed to support various features, e.g., networking, connectivity (IoTs), and multimedia support).

☐

The problem of the simple batch monitors is that a machine should wait while a job processes I/O operations (e.g., reading data from magnetic tapes). To solve this problem, in the late 60s, multi-programmed batch monitors were proposed.



In the 60s - 90s, humans were expensive and computers became cheap; thus, people connected multiple computers and let the OS manage the cluster.



In the 60s, to reduce the human labor, OSes worked as a simple batch monitor. OSes load multiple jobs written in magnetic tapes and process them one at a time.

Question 7

2 / 2 pts

What are the three major functionalities of operating systems?

☒ Provide an abstraction for the complex underlying hardware operations

☐ Secure user applications from external cybersecurity threats

☒ Manage system resources (e.g., CPUs, memory or disk storage)

☒ Provide standard interfaces to user applications



Run ready-made applications, e.g., web servers, to facilitate users' applications

Question 8

1 / 1 pts

Multi-programmed means multiple programs are running in parallel (simultaneously)



True

☒ False

Question 9**1 / 1 pts**

Multi-processing means multiple processors are; thus, we can run jobs in parallel

☒ True

☐ False

Question 10**1 / 1 pts**

Multi-threading means multiple threads can run in parallel (simultaneously)

☐ True

☒ False

Question 11**1 / 1 pts**

Linux is an open-source operating system, while POSIX is OS standards specified by IEEE

☒ True

☐ False

Question 12**1 / 1 pts**

Linus Torvalds lived in Oregon five years ago, and now lives in California

- ☐ True
- ☒ False

Question 13**2 / 2 pts**

Which of the following statements is false?

- ☐ OS defines a process using the process context block (or process context structure)
- ☐ Each process has its own memory address space.
- ☐ In C, local variables are stored in the stack segment.
- ☐ In C, malloc allocates memory in the heap segment.
- ☒ Stack may suffer from memory fragmentation issues, while heap is not

Question 14**2 / 2 pts**

What are the segments composing a process (choose all that apply)?

- ☒ Heap segment (i.e. heap memory)

- ☐ CPU register segments
- ☒ Code segment (i.e., instructions)
- ☒ Stack segment (i.e., stack memory)
- ☒ Data segment
- ☐ File segment (i.e., data in disks)

Question 15**1 / 1 pts**

Process is an abstract view of a program running on OS

- ☒ True
- ☐ False

Incorrect**Question 16****0 / 1 pts**

Process ID is guaranteed to be unique and won't be changed until we reboot an OS

- ☒ True
- ☐ False

Question 17**1 / 1 pts**

fork() returns the child PID in the calling process, while it returns the parent PID in the child

☐ True

☒ False

Incorrect

Question 18

0 / 1 pts

execvp() creates a child process by fork() and dump the executing program to the child

☐ True

☒ False

Question 19

2 / 2 pts

What is not true about stack and heap memory (choose all that apply)?



If a process create two threads(), they have their own stack and heap memory



OS defines five thread states: new, ready, running, blocked, and terminated



If a process fork(), the child will share the parent's stack and the heap memory



OS only allocates memory in heap; user programs cannot allocate heap memory



Memory fragmentation can occur in Stack, but cannot occur in Heap

Question 20

2 / 2 pts

What are the scheduling states a process can have in the OS (choose all that applies)?

☐ Transition

☒ Ready

☒ Waiting

☐ Interrupted

☐ Blocked

☐ Zombie

☒ New

☒ Terminated

☒ Running

Incorrect

Question 21

0 / 2 pts

Which of the following statements is true?

☐

Multiple threads in a process can't share static variables unless it's protected by mutex

☐

Like processes, a thread can have four states (i.e., new, ready, running and blocked)

☐

It's required to explicitly call pthread_exit() to terminate a thread.

☒

In the fork-join pattern, we call pthread_join() with all the running threads' IDs wait until all of them are terminated execution.

☒

Like processes, threads contain the four segments.

Quiz Score: **30** out of 34