Pre-Lecture 11

Due Sep 30 at 9am **Points** 11 **Questions** 8 **Available** until Sep 30 at 9am
Time Limit None **Allowed Attempts** 2

Instructions

Take this quiz *after you have watched the required videos and/or read the associated sections of the textbook*. See **Lecture 11: Processes I**.

You may attempt this quiz twice. Incorrect responses are marked after each attempt. Correct answers are revealed at the start of class for this lecture.

Carefully note the deadline for responses. Submissions are not accepted after the deadline, and there is no grace period.

This quiz was locked Sep 30 at 9am.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 2	7 minutes	10.33 out of 11
	Attempt 1	3,710 minutes	10.67 out of 11

Score for this attempt: **10.33** out of 11
Submitted Sep 30 at 8:58am
This attempt took 7 minutes.

Question 1 — 1 / 1 pts

Which of the following scenarios qualify as exceptional control flow for a program? (Select all that apply.)

- [ ] calling a sort procedure
- [x] data arriving from the network — Correct!
- [x] divide-by-0 error — Correct!
- [x] expiration of a timer — Correct!
- [ ] for loop

- [ ] switch statement

Question 2

0.33 / 1 pts

Suppose that the user code incurs an exception *k* at an instruction. Which of the following are ways in which the kernel code's handler for exception *k* may cause the user code to resume execution? (Select all that apply.)

- [ ] Execution of the user code resumes by restarting at the first instruction.
- [x] Execution of the user code resumes by retrying the current instruction. — Correct!
- [ ] Execution of the user code resumes at the next instruction. — orrect Answer
- [ ] Execution of the user code does not resume at all. — orrect Answer

Question 3

2 / 2 pts

Match the scenario on the left with the kind of exception on the right. Some scenarios may match more than one kind of exception; however, you are to use each of the four kinds of exceptions exactly once.

Correct! **This exception is handled by aborting the user code.**	abort
Correct! **After this exception is handled, the user code resumes at the next instruction.**	trap
Correct! **This exception is not related the program in execution.**	interrupt
Correct! **After this exception is handled, the user code resumes at the current instruction.**	fault

Question 4

1 / 1 pts

Each process has its own stack pointer register.

- True
- Correct! False

Question 5

1 / 1 pts

Suppose that you accidentally executed an infinite-loop program, but no longer have access to the command line / terminal from which you ran the program. Upon opening another terminal, you type command 'ps a' and see this:

```
  PID TTY      STAT   TIME COMMAND
 9551 pts/1    Ss     0:00 -tcsh
 9575 pts/0    S+     0:00 ./a.out
 9577 pts/1    R+     0:00 ps a
```

What *exact* command should you type to stop execution of the infinite-loop program?

Correct!

kill 9575

orrect Answers

kill 9575

kill -9 9575

Question 6

1 / 1 pts

Consider the following C program:

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

int main() {
  printf("%d\n", getppid());
```

```
  while(1) {}
}
```

Suppose that we execute the program *in the background* with the command './a.out &'. This gives us the ability to run the ps command while the C program is executing. The following is displayed by the ps command:

```
  PID TTY      STAT   TIME COMMAND
 9489 pts/0    Ss     0:00 -tcsh
10314 pts/0    R      0:14 ./a.out
10320 pts/0    R+     0:00 ps a
```

What is printed by the C program?

Correct!

9489

orrect Answers

9489

Question 7

1 / 1 pts

How many times is "Hello, world" printed by the following C program?

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

int main() {
  fork();
  fork();
  fork();
  printf("Hello, world\n");
}
```

Correct!

8

orrect Answers

8

eight

Question 8

3 / 3 pts

Consider the following C program:

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

int main() {
  int pid1;
  int pid2;
  int pid3;
  int pid4;

  pid1 = getpid();
  pid2 = fork();
  pid3 = fork();
  pid4 = getppid();

  if(pid1 == pid2)
    printf("u");
  if(pid1 == pid3)
    printf("v");
  if(pid1 == pid4)
    printf("w");
  if(pid2 == pid3)
    printf("x");
  if(pid2 == pid4)
    printf("y");
  if(pid3 == pid4)
    printf("z");

  sleep(1);   // simply pauses execution, does not affect what is printed
}
```

One possible output of program: wwx

How many different outputs are possible? 3

Notice that there are no spaces or newlines printed.

HINT: Draw the process graph.

Answer 1:

Correct! wwx

orrect Answer xww

orrect Answer wxw

Answer 2:

Correct! 3

orrect Answer three

Quiz Score: **10.33** out of 11