

Quiz: The Process Concept

Total points **92/93**

Take the quiz solo, but feel free to consult a partner student, the book, the videos or other resources if needed. Re-take quiz if your score is less than 80% or if you just want some more practice.

The respondent's email (**faiyaz@pdx.edu**) was recorded on submission of this form.

✓ How many processes can an OS support at one time? * 5/5

- ☐ 42
- ☐ One more than the number of CPUs
- ☐ Exactly the same number as the number of CPUs
- ☒ A large number of processes, many more than the number of CPUs ✓

✓ A program is a single instance of a process * 5/5

- ☐ True
- ☒ False ✓

✓ A process is a program in execution * 5/5

- ☒ True ✓
- ☐ False



✓ A process has current "state", including memory values, register values, open files, etc. *5/5

☒ True ✓

☐ False

✓ Is a segmentation fault an example of limited direct execution? * 5/5

☐ No, it is a fault not an interrupt

☐ Yes, concurrent access to I/O devices is not allowed

☐ No, seg faults are caused by programmer error

☒ Yes, the MMU limits which memory addresses can be accessed by a program ✓

✓ How do system calls help to implement limits on direct execution? * 5/5

☒ system calls allow the OS to expose a limited set of functionality to application programs ✓

☐ by allowing the OS to run before and during the application's request. the OS can then enforce whatever limits it needs to.

☐ system calls are invoked mainly via timer interrupts which may not be altered by the application programmer

☐ by handling faults for such problems as divide by zero

☐ because there are a limited number of system calls



✓ Each process has its own virtual memory address space. *

5/5

☒ True



☐ False

✓ The act of transitioning from one running process to another is often called what?

*5/5

☐ resume

☒ context switch



☐ state suspension

☐ swapaholism

☐ temporary suspension



Which of the following are part of "process state", i.e., the information that an OS ^{*} maintains for each process.

	Yes, part of process state	No, not part of process state	Score	
location of data space (the heap)	<input checked="" type="radio"/>	<input type="radio"/>	1/1	✓
stack location	<input checked="" type="radio"/>	<input type="radio"/>	1/1	✓
register values	<input checked="" type="radio"/>	<input type="radio"/>	1/1	✓
a list of functions that have been called by the program	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
a current working directory within the file system	<input checked="" type="radio"/>	<input type="radio"/>	1/1	✓
programming bugs	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
political affiliation (red states vs. blue states)	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
whether or not the process is confused	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓



✓ On a single CPU (a single core), how many user-level processes are active *5/5 at any one time?

- ☐ depends on the programming language used by the applications
- ☐ none
- ☒ at most one ✓
- ☐ there is no limit
- ☐ one

✓ When a process is running but then it needs to wait for an I/O operation *5/5 to complete, the OS transitions the process to which state?

- ☐ ready
- ☐ running
- ☒ blocked ✓

✓ When a process is in the blocked state and then it's waited-for I/O *5/5 operation completes, the OS transitions the process to which state?

- ☐ running
- ☒ ready ✓
- ☐ blocked



✓ When a process is in the ready state and the OS decides to schedule/run this process, then the OS transitions the process to which state? *5/5

☒ running ✓

☐ ready

☐ blocked

✓ If a process runs long enough to use up its entire time slice, and a timer interrupt occurs (signalling the end of the time slice) then what state will the OS transition this process to? *5/5

☐ blocked

☐ running

☒ ready ✓



Of the following ways for processes to terminate, which of them are voluntary (i.e., the process itself decided to terminate) vs. involuntary (something other than the process itself initiates the termination).

*

	voluntary termination	involuntary termination	Score	
normal, successful exit()	<input checked="" type="radio"/>	<input type="radio"/>	1/1	✓
terminated by the OS to free up resources	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
process calls exit() indicating that the program detected an error	<input checked="" type="radio"/>	<input type="radio"/>	1/1	✓
segmentation fault causes the process to crash and terminate	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
parent process terminates child process	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
human types Ctrl-C (or other suitable action) to terminate the process	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓



✓ In Linux/Unix which system call causes a new process to be created as a child of the current process? *5/5

☒ fork() ✓

☐ exec()

☐ createprocess()

☐ wait()

✓ In Linux/Unix, immediately after a process is created, what does it share with its parent process? *5/5

☐ instructions/code

☐ register values

☐ file descriptors

☐ memory contents

☒ none of the above. the child process has copies of these items but does not actually share any of them with its parent ✓



In Linux/Unix, immediately after a process is created, what does it share with its parent process? *

	yes, child shares this with the parent	no, child has its own copy of this	Score	
instructions/code	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
register values	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
file descriptors	<input type="radio"/>	<input checked="" type="radio"/>	1/1	✓
memory contents	<input checked="" type="radio"/>	<input type="radio"/>	0/1	✗

This form was created inside of Portland State University.

Google Forms

