

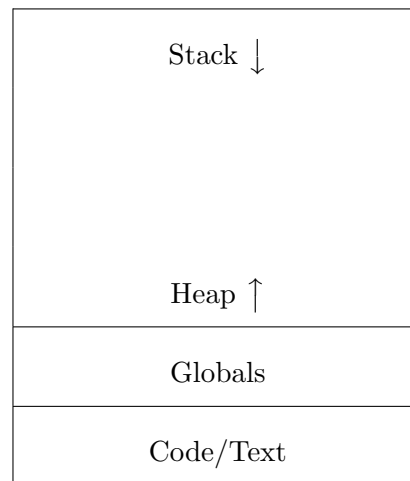
CSSE 332 -- OPERATING SYSTEMS

Introduction to Processes

Name:

SOLUTION KEY

Question 1. (5 points) The figure below represents the address space of a process P. Label each part of the process address space with its corresponding section content (i.e., stack, heap, etc.).

High Address**Low Address**

Question 2. (5 points) Describe how processes are related to each other in a Unix-like operating system.

Solution: There is one mother process (typically called `init` or depending on the system, `systemd`) that is initially created. All processes in the system are children (direct or indirect) of this process. It is like a big tree rooted at this `init` process.

Question 3. (5 points) How does a process keep track of who its direct parent is?

Solution: We maintain all of the information about a process in the process control block (pcb). In xv6, it is called the `struct proc` and is defined in `kernel/proc.h`. This is where we also store the process's trapframe, it is maintained in the kernel's memory space.

Question 4. (5 points) In the standard C library, the `fork` system call is used to create a new process by duplicating the calling process. The new process is called a child of the calling process. Finally, a process can use the `getpid` system call to obtain its process id.

Question 5. (a) (5 points) Where can you find the documentation for the `fork` system call? What is the command you can use to bring it up?

Solution: The manual pages, using `man fork`.

(b) (5 points) From the documentation page, which header file should you include to use `fork`?

Solution: `unistd.h`

Question 6. (5 points) Consider the code snippet below.

```
1 pid_t pid = fork();
2 if(pid == 0) {
3     printf("Hello from the child process %d\n", getpid());
4     exit(0);
5 } else {
6     printf("Hello from the parent process %d\n", getpid());
7     exit(0);
8 }
```

Which of the print statements will show up on the console first?

Solution: We cannot know since we do not control which process runs at which time, even if we use `fork`.

Question 7. (5 points) Consider the code snippet below.

```
1 pid_t my_pid = getpid();
2 if(fork() == 0) {
3     printf("My pid is %d\n", my_pid);
4     exit(0);
5 } else {
6     printf("My pid is %d\n", my_pid);
7     exit(0);
8 }
```

Which of the following statements is **True**?

A. Each process will print its own process id.

- B. Both processes will print the same value, which is the process id of the parent.**
- C. Both processes will print the same value, which is the process id of the child.
- D. We cannot know what values will be printed in each case.
- E. None of the above.

Question 8. (10 points) Consider the following snippet of code.

```
for(int i = 0; i < 3; i++)  
    fork();
```

How many process will we end when this loop runs? Draw the corresponding tree of these processes.

Solution: We would end up with 2^3 processes since each new process will continue with the loop.