CS 330 & CS 332 Midterm 2 Prep

UNIX and Assembly Questions

TRUE/FALSE

1. You can use "man" pages to learn more about programs, utilities, and functions

1. You can use "man" pages to learn more about programs, utilities, and functions

TRUE

The man pages (manual pages) are documentation for commands, calls, and the like. They can be accessed by typing "`man <command>`" at the command line.

2. In C compilation, the Assembler converts C code (.c) into Assembly code (.s).

2. In C compilation, the Assembler converts C code (.c) into Assembly code (.s).

FALSE

The naming is misleading, but the Assembler converts Assembly code (.s) into Object code (.o).

3. `strtok()` will always tokenize a given string on any whitespace it encounters.

3. `strtok()` will always tokenize a given string on any whitespace it encounters.

FALSE

`strtok()` expects a string (char array) and a delimiter (char).

4. The stack follows the principle of last in first out.

4. The stack follows the principle of last in first out.

TRUE

Elements are added to and removed from the top of the stack.

5. `strdup()` is used to duplicate a string.

5. `strdup()` is used to duplicate a string.

TRUE

Upon success, `strdup()` returns a pointer to the duplicated string.

6. We pop data onto a stack to add it and push data off it to remove it.

6. We pop data onto a stack to add it and push data off it to remove it.

FALSE

The opposite is true! We push data to the top of the stack to add it and pop data off the top of the stack to remove it.

7. User Mode and System Mode are essentially the same mode of execution.

7. User Mode and System Mode are essentially the same mode of execution.

FALSE

The key difference is that user mode does not have access to system resources, while system mode does. Therefore, user mode is a less privileged mode of execution than system mode (aka kernel or control mode).

8. In Assembly, the stack pointer `%rsp` holds the address of the top stack element.

8. In Assembly, the stack pointer `%rsp` holds the address of the top stack element.

TRUE

9. `fork()` creates a new process by cloning the calling process.

9. `fork()` creates a new process by cloning the calling process.

TRUE

10. In Assembly, you can repeat sections of code using while and for loops.

10. In Assembly, you can repeat sections of code using while and for loops.

FALSE

To implement a "loop" in Assembly, we use conditional jumps to repeat sections of code.

11. The `exec()` library of functions replace the current process image with a new process image.

11. The **`exec()`** library of functions replace the current process image with a new process image.

TRUE

This is exactly what exec() functions do!

12. In Assembly, we navigate the stack by incrementing and decrementing the stack pointer.

12. In Assembly, we navigate the stack by incrementing and decrementing the stack pointer.

TRUE

13. Zombie processes are an inevitable side effect of process spawning.

13. Zombie processes are an inevitable side effect of process spawning.

FALSE

We have whole libraries of functions that can be used to prevent this, such as `wait()` and `SIGNAL` commands.

14. Recursion is impossible in Assembly.

14. Recursion is impossible in Assembly.

FALSE

You can absolutely implement recursion in Assembly!

15. The `jobs` command will show a list of current jobs along with their state, number, and name.

15. The `jobs` command will show a list of current jobs along with their state, number, and name.

TRUE

MULTIPLE CHOICE

16. How are getline() and getdelim() different?

A. getdelim works like getline, except you can specify a delimiter other than the "\n" character.
B. They are not different functionally.
C. getdelim returns all the instances of a specified delimiter.
D. getdelim works like strtok, while getline does not.

16. How are getline() and getdelim() different?

A. getdelim works like getline, except you can specify a delimiter other than the "\n" character.
B. They are not different functionally.
C. getdelim returns all the instances of a specified delimiter.
D. getdelim works like strtok, while getline does not.

17. Which of the following would you use to implement a switch case in Assembly?

A. conditional moves
B. while loops
C. stack trace
D. jump table

17. Which of the following would you use to implement a switch case in Assembly?

A. conditional moves
B. while loops
C. stack trace
D. jump table

18. A process can be defined as which of the following?

A. A program in execution.
B. An instance of a running program.
C. The entity that can be assigned to, and executed on, a processor.
D. A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources.
E. All of the above.

18. A process can be defined as which of the following?

A. A program in execution.
B. An instance of a running program.
C. The entity that can be assigned to, and executed on, a processor.
D. A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources.
E. All of the above.

19. What is the return command in Assembly?

A. return
B. ret
C. exit
D. pop

19. What is the return command in Assembly?

A. return
B. ret
C. exit
D. pop

20. What is a kernel?

A. A firewall that protects computer hardware.
B. An API which grants the user access to the computer's lowest level resource management.
C. The core of a computer's OS that facilitates interactions between software and hardware.
D. A tasty seedling found on ears of corn.

20. What is a kernel?

A. A firewall that protects computer hardware.
B. An API which grants the user access to the computer's lowest level resource management.
C. The core of a computer's OS that facilitates interactions between software and hardware.
D. A tasty seedling found on ears of corn.

21. How do you run a program in the background?

A. a.out bg
B. & a.out
C. bg a.out
D. a.out &

21. How do you run a program in the background?

A. a.out bg

B. & a.out

C. bg a.out

D. a.out &

22. wait() does which of the following?

A. Pauses execution of a program for a number of seconds specified by the user.
B. Waits for a child process to change state.
C. Pauses a program until it receives input from the user.
D. Waits for a function to return successfully.

22. wait() does which of the following?

A. Pauses execution of a program for a number of seconds specified by the user.
B. Waits for a child process to change state.
C. Pauses a program until it receives input from the user.
D. Waits for a function to return successfully.

23. What is the difference between execl() and execv() functions?

A. execl() accepts a list of values while execv() accepts an array of values.
B. execv() accepts a list of values while execl() accepts an array of values.
C. There is no difference between them.
D. execl() only works on Linux while execv() will work on all systems.

23. What is the difference between execl() and execv() functions?

A. execl() accepts a list of values while execv() accepts an array of values.
B. execv() accepts a list of values while execl() accepts an array of values.
C. There is no difference between them.
D. execl() only works on Linux while execv() will work on all systems.

24. Processes in a Linux environment can exist in which of the following states?

A. foreground
B. background
C. suspended
D. All of the above.

24. Processes in a Linux environment can exist in which of the following states?

A. foreground
B. background
C. suspended
D. All of the above.

Thank you for coming!

Please write your BlazerID on the whiteboard before you leave.