

# CS3560: Exam 2

THURSDAY, JUNE 5, 2003

Instructor: Andrew Prock  
prock@mcs.csu Hayward.edu

NAME:

PLEASE MAKE SURE YOU FOLLOW ALL INSTRUCTIONS!

**Read all instructions carefully!**

- A copy of the test may be found on the course web-page.
- The exam is due by midnight Thursday June 5th 2003.
- The exam may be turned in either electronically, or physically to the department office, where it will be placed in my mailbox.
- If you turn in a hard copy of the exam, please attach any additional work using a secure method, such as a stapler or folder.
- There are strict word limits on all essay questions. If you exceed these limits you will lose 1 pt per word you run over.
- You may use any resource other than your text and class notes to complete this exam.
- You will not answer all questions. Some questions are optional. Other questions are to be answered by particular sets of students. Make sure you understand which questions you are responsible for.
- Some questions may be poorly worded, unclear, or ambiguous. In these situations, select what you think is the **single best** answer for the question.
- There are 200 points total, with no bonus.
- Answer all questions to the best of your ability. Partial credit will be given, so do your best on each. If you are unable to answer a question completely for some reason, try to explain in words what needs to be done.
- Good luck!

1. (70 pts, 1 per) True or False

- (1) A file may have multiple read locks.
- (2) A file may have multiple write locks.
- (3) You can use `mmap` to write data to a file.
- (4) Every signal will cause a process to exit unless handled.
- (5) To load a new program you must call `fork`.
- (6) To work properly `setjump` and `longjump` must be called from within the same scope.
- (7) Both `exit` and `_exit` close all open files.
- (8) A single `fork` can create up to three new processes.
- (9) Pipes are best used with single process programs.
- (10) A zombie process requires no kernel resources.
- (11) `pause` pauses execution of a parent process till a child process terminates.
- (12) `vfork` is the same as `fork` except that it doesn't copy the parent's process id.
- (13) I/O efficiency is independent of the number of kernel calls that are made.
- (14) Every signal has a default signal handler.
- (15) Every signal can be ignored.
- (16) To work properly `setjump` and `longjump` must be called from within the same scope.
- (17) You can use `mmap` to write data to a file.
- (18) Every signal will cause a process to exit unless handled.
- (19) Pipes are best used with single process programs.
- (20) File locks are stored in the v-node table.
- (21) A program can execute either `exit` or `_exit`, not both.
- (22) `SIGSTOP` indicates that a process has been suspended (using control Z)
- (23) The `kill` function allows a process to send a signal to itself.
- (24) The file offset must refer to a location within the file.
- (25) A function which can safely call itself is `reentrant`.
- (26) The library function `calloc` is for object allocation, not byte allocation.
- (27) The `sigaction` function is used to register a handlers for all the signals.
- (28) `sigsuspend` is atomic as `longjmp` is to race conditions.
- (29) The `volatile` keyword is used to describe variables which are stored in registers.
- (30) After a `fork`, the parent and child processes have the same value for the file offsets stored in their process memory.
- (31) The functions `sigprocmask`, `sigaddset`, and `sigpending` are all used to interact with the current process mask.

- (32) After an `exec` the process will be assigned a new process id.
- (33) The `raise` function sends a signal.
- (34) It is possible to simulate multiple alarms with a single `ALARM` signal.
- (35) Race conditions would still exist without signals.
- (36) POSIX defines a right standard which guarantees porting will be trivial.
- (37) When `exec` is called, all leaked memory is restored.
- (38) If a call to `wait` returns, then a child process has terminated.
- (39) When a program core dumps, `_exit` is called, leaving some file descriptors open.
- (40) If there is a process holds a read lock on a byte, it can also be granted a write lock.
- (41) All library calls make system calls.
- (42) `getc` is to `getchar` as `putc` is to `putchar`.
- (43) Every atomic system call is atomic.
- (44) `O_TRUNC` is a flag used with the `fopen` function.
- (45) The Standard I/O library is reentrant.
- (46) The execution of a program does not start in `main`
- (47) After a fork occurs, any alarm set in the parent will also be set in the child.
- (48) The mode of a file stores its access permissions (user/group/other +/- read/write/execute).
- (49) The stack and the heap are in different process spaces.
- (50) File locks set by the parent are not inherited by the child process after a fork.
- (51) If a program is terminated abnormally by some signal, a core dump might be produced.
- (52) A regular link can refer to files across partitions
- (53) The zeroth argument of a spawned program is the name of the program.
- (54) In the memory layout of a program, all initialized variables are stored below the heap.
- (55) For every call to `fwrite` there is a call to `write`
- (56) The offset argument to `lseek` must be positive.
- (57) `sync` is to kernel memory as `fflush` is to user memory.
- (58) There can be two entries in the v-table for the same file.
- (59) When a `write` is done on a file opened in append mode, that write is atomic.
- (60) The `vfork` function does not clone the parent address space.
- (61) Pipes behave like normal files with respect to system I/O
- (62) Every constant defined in the POSIX standard can be found in the appropriate header files.
- (63) Due to the size limitations of the `int` type, there can be no more than 32 signals.
- (64) The `sleep` function must use the `SIGALARM` signal to work properly.

- (65) When processes share a common resource, process synchronization is necessary.
- (66) When a signal handler is interrupted by another signal handler, if both are reentrant, no race condition can exist.
- (67) `malloc` and `calloc` leave memory uninitialized.
- (68) Ownership of a lock is stored with the lock.
- (69) For every call to `new` there is a call to `delete`
- (70) If a process registers a signal handler for SIGINT, and that signal occurs twice, the handler will be called twice.

2. (15 pts, 1 per) Multiple choice.

- (1) When a child's parent process terminates, which of the following is not true
  - (a) the child will be inherited by the `init` process
  - (b) the parent process sends a `SIGCHLD` signal
  - (c) the exit status of the parent is stored till the child calls `wait`
  - (d) the parent may become a zombie process
- (2) Which of the following are not true of `execve`?
  - (a) it replaces the stack segment of the process.
  - (b) it replaces the data segment of the process.
  - (c) it replaces the file locks of the process.
  - (d) it replaces the environment of the process.
- (3) What does `flush` do?
  - (a) delete everything in the buffer
  - (b) write everything in the buffer to the standard I/O
  - (c) copy the buffer contents using `write`
  - (d) copy the string stored in the buffer into the kernel
- (4) Which of the following are not in the directory block?
  - (a) i-node number
  - (b) filenames
  - (c) pathnames
  - (d) the current directory
- (5) When the `lstat` function is called a `struct stat` is filled in using information stored in.
  - (a) the kernel
  - (b) the i-node
  - (c) the directory block
  - (d) the file table
- (6) Which of the following can never be considered a slow system call.
  - (a) I/O to a pipe
  - (b) I/O to a disk file
  - (c) I/O to a network file
  - (d) closing a disk file
- (7) The default handler for which of the following signals does not cause the process to terminate:
  - (a) `SIGABRT`
  - (b) `SIGALRM`
  - (c) `SIGSTOP`
  - (d) `SIGUSR1`

- (8) When a signal occurs a program can tell the kernel it wants to:
- (a) Ignore the signal
  - (b) Catch the signal
  - (c) Allow the default action to occur
  - (d) All of the above
  - (e) None of the above
- (9) Which of the following locks will be granted?
- (a) A write lock for a file with a write lock.
  - (b) A write lock for a file with a read lock.
  - (c) A read lock for a file with a write lock.
  - (d) A read lock for a file with a read lock.
- (10) Pipes can be used to:
- (a) output data to a file.
  - (b) share file table entries between processes.
  - (c) communicate between processes.
  - (d) control how the kernel schedules two processes.
- (11) Which of the following is true about file I/O.
- (a) Efficiency is only dependent on the file size.
  - (b) `mmap` is always more efficient.
  - (c) The standard I/O library is independent of the system calls.
  - (d) Efficiency is dependent on the number of kernel calls made.
- (12) Which of the following tables is not used by the kernel to manage file sharing?
- (a) The v-node table
  - (b) The i-list table
  - (c) The file table
  - (d) The process entry table
- (13) Which of the following are wholly independent of the `fcntl` function
- (a) the `exec` family of functions.
  - (b) the `fork` function
  - (c) the `write` function
  - (d) the `close` function
- (14) Which of the following information is not stored in the kernel
- (a) process table entries
  - (b) file table entries
  - (c) v-node table entries
  - (d) file lock information

- (15) Do you know where your forked children are? Note: This program uses the TELL and WAIT functions in the text. They could be the signals or the pipes implementation, it doesn't matter.

```
int main (int argc, char **argv) {

    pid_t pid;
    char c[33] = "Hi! This is a useless program!\n";
    int status;

    TELL_WAIT();

    pid = fork();

    if (pid == 0) {
        printf("%s", c);
        TELL_PARENT(getppid());
    }
    else {
        // Wait for the child process
        WAIT_CHILD();
        printf("%s", c);
        pid = getpid();
    }

    printf("Goodbye, this has been %d.\n", pid);
}

// What is the difference in output between a line-buffered
// and fully-buffered stdout? Here's a possible test run:
// % a.out
// Hi! This is a useless program!
// Hi! This is a useless program!
// Goodbye, this has been 3244.
// Goodbye, this has been 3243.
```

- (a) There's no difference.
- (b) The line-buffered device will not output as above.
- (c) The fully-buffered device will not output as above.
- (d) It depends on the system.
- (e) There is a bug that affects this.

3. (30 pts) Long Essay

Write a brief essay (100 words or less) which addresses one of the following topics

- (option a) Describe some system programming that you have done outside of this class, and discuss how it relates to the topics that were covered.
- (option b) Explain how organization of the book is related to the organization of the UNIX system. In particular, discuss the the order of topic presentation, as well as topic grouping.
- (option c) Describe how the information and topics covered in this course might be useful to people who are not system programmers. Use at least two concrete examples of situation where this knowlege would provide practical use.

4. (50 pts, 10 per) Short Essays

If your first name begins with a letter between A-G answer the following set of essay questions.

- (1) In 25 words or less, describe the difference between a system call and a library call.
- (2) In 25 words or less, describe the difference between UNIX and POSIX.
- (3) In 25 words or less, explain why idioms like `"_t"` and the "preceding underscore" are important.
- (4) In 25 words or less, describe all occasions when the Standard I/O library flushes it's buffers.
- (5) In 25 words or less, explain why process masks – which are used to block signals, are necessary.

If your first name begins with a letter between H-R answer the following set of essay questions.

- (1) In 25 words or less, explain why having two process write to the same file can cause problems.
- (2) In 25 words or less, explain what a system programmer does.
- (3) In 25 words or less, describe the costs associated with making a system call.
- (4) In 25 words or less, describe the information which is passed to a newly loaded program.
- (5) In 25 words or less, explain why `sigsetjmp/siglongjmp` may cause problems when used inside of a signal handler.

If your first name begins with a letter between S-Z answer the following set of essay questions.

- (1) In 25 words or less, describe the difference between an inode and a file.
- (2) In 25 words or less, explain how the kernel loads a program.
- (3) In 25 words or less, explain how a child process differs from a parent process.
- (4) In 25 words or less, explain how a `sigset_t` is different from a process mask.
- (5) In 25 words or less, discuss how "Murphy's Law" relates to race conditions.

5. (35 pts) Programming exercise

You are to do an exercise from the book. For this question, you will need to summarize your results by answering the questions that the text asks. You must include both your source code as well as the written explanation for the exercise.

For your written explanation, limit the total number of words to 50 words or less.

If your **LAST** name begins with a letter between A-L do programming exercise 10.10 from the text.

If your **LAST** name begins with a letter between M-Z do programming exercise 10.12 from the text.

**READ THIS STATEMENT AND SIGN BELOW:**

I have read over this exam, if time permitted me, and have completed it to the best of my independent ability. I have adhered to the instructions listed on the cover sheet of this exam. I have looked at and am turning in all 10 pages of the exam (counting and including this page).

SIGNED: