**<p>** A hard link allows a file too have more than one name, but the names can’t be in the same directory.,False **<p>** A hard link can only be used to link files within the same directory.,False **<p>** A process can create a child with new environment variables by constructing an appropriate argument for the "fork" system call.,False **<p>** A program writing a lot of data to a tape drive would normally choose a block special device rather than a character device, since higher speed can be achieved by avoiding transferring data directly from the user's' address space to the tape.,False **<p>** A program writing a lot of data to a tape drive would normally choose a character special device rather than a block device, since higher speed can be achieved transferring data directly from the users buffer to the tape.,True **<p>** A symbolic link allows a path name residing on one disk to actually refer to a file on another disk.,True **<p>** A True **<p>** multitasking operating system requires an MMU to ensure that each process has its address space protected from corruption by another process.,True **<p>** A typedef statement in C that defines a struct, allocates the actual storage needed to accommodate all the fields of that struct.,False **<p>** After an exec system call the child process signal table has the same values at the one in the parent.,False **<p>** After an exec system call the signal table that determines how to deliver each signal is unchanged.,False **<p>** Although it is possible to have more than 1 Linux pathname correspond to the same inode, the two pathnames cannot end in the same directory.,False **<p>** An important advantage of the STDIO library is that it provides a convenient buffering mechanism that avoids unnecessary system calls.,True **<p>** By default Linux uses synchronized I/O for disk writes.,False **<p>** By default output to an ordinary file done with the Linux "write" system call is copied to the buffer cache before the system call returns. By using fcntl to manipulate the O\_SYNC flag, that behavior can be changed to cause the "write" system call to return before that copy is complete.,False **<p>** By using an MMU, the operating system could put each process's "main" at the same logical address.,True **<p>** Data transfers with pipes, fifo's and message queues all require two system calls, one for the sender to transfer the data to the system's BSS and another to copy it back to the other process.,True **<p>** Doing 1 byte "write"s will be slow because of all the extra disk activity.,False **<p>** Each a.out file contains an ELF header that establishes the size of the TEXT, DATA, and STACK segments of the file, and optionally information on the symbol table.,False **<p>** File descriptors are copied after an EXEC and while valid after an EXEC have offsets reset to 0.,False **<p>** File descriptors are stored in a processes "System Data Segment" (Ublock) and thus are inherited by a child subprocess after fork.,True **<p>** File descriptors are stored in a processes "System Data Segment" (UBLOCK) and thus are still valid after an exec.,True **<p>** For programs with a big text segment, an MMU provides great savings in implementing the "fork" system call.,True **<p>**. If the file corresponding to an open file descriptor is removed, subsequent reads from that file descriptor are still valid.,True **<p>** If the file corresponding to an open file descriptor is removed, subsequent reads from that file descriptor will show end of file.,False **<p>** In C, space acquired with "malloc" and assigned to a pointer will eventually be garbage collected when no pointers refer to it.,False **<p>** In a pthreads program every mutex must be associated with a condition variable.,False **<p>** It is important that a parent process do nothing but "wait" for its children to terminate. Any use of its global variables may have unpredictable effects on its children.,False **<p>** It is not possible for a process to alter an environment variable for its parent.,True **<p>** It is possible for a process to successfully issue an "exec" system call, without ever having issued a "fork" system call.,True **<p>** Linux allocates a semaphore for each special device file to be sure that no two processes can use them at once,,False **<p>** Linux can detect the end of a text file since all text files end in a NULL byte.,False **<p>** Linux imposed a "type" on each ordinary file, by writing a "magic #" in the first two bytes of the file.,False **<p>** Linux keeps track of the length of each of its ordinary files by a field in it's inode. However, each file must have an even length and is padded with a blank if necessary.,False **<p>** Linux keeps track of the size of each of its directory files by a field in its inode.,True **<p>** Linux programs are passed implicit information in environment strings located in the DATA segment.,False **<p>** Linux uses a "super block" on each disk to describe parameters of the filesystem on that disk,True **<p>** Normally, programs returning from a "write" system call should not immediately alter the data in the supplied buffer since the OS may still be copying it to the buffer cache.,False **<p>** One advantage of using the C language for systems programming is that it has very powerful operators for input/output built into the language itself.,False **<p>** Pipes transfer data more quickly than Message Queues, since the data is transferred directly between processes without using storage in the operating system.,False **<p>** Pointer variables in C are always allocated in the BSS.,False **<p>** Programs that use the "read" system call to sequentially read 1 byte at a time will run slowly because of all the disk activity this causes.,False **<p>** Since system calls are expensive, a very good way to process a very large file is to get its size with "stat", malloc a buffer of that size, and read the file with one system call.,False **<p>** The "break 100" command in gdb, causes the debugger to stop immediately after executing the statement at line 100.,False **<p>** The "break" command in gdb, causes the debugger to stop immediately after executing the statement at the line number specified.,False **<p>** The data structure “aiocb” is used to force write system to calls to block until the data reaches the disk.,False **<p>** The "fcntl" system call, allows a program to alter the flags set during "open", without having to first close the file,True **<p>** The "fcntl" system call, is the only way a program can get non-blocking behavior from a pipe.,True **<p>** The "fflush" function allows a program to be sure all data that has been written has reached the disk drive.,False **<p>** The "fsync" system call, allows a program to alter the flags set during "open", without having to first close the file.,False **<p>** fsync - synchonize a file's in-core state with storage device. Flushes all buffer cache pages of fd to disk device. Doesn't necessarily ensure entry in directory containing the file has also reached disk.,True **<p>** The "printf" operator like any function in C, is simply a library routine, which runs when the a.out file is executed. The C compiler has no involvement in this computation.,True **<p>** The "shmat" system call does not actually allocate new memory. It just changes some entries in the MMU.,True **<p>** The "sizeof" operator in C provides a convenient way to determine how large a file is without using the "stat" system call.,False **<p>** The "sizeof" operator in C provides a convenient way to tell the size of a file.,False **<p>** The "sizeof" operator like any function in C, is simply a library routine, which runs when the a.out file is executed. The C compiler has no involvement in this computation.,False **<p>** The C compiler uses the STACK segment to assign storage to all variables declared within the body of a function, with the possible exception of those declared with storage class "static" or "register".,True **<p>** The C language has no particular built in support for Linux system calls. It could be used equally well for systems programming on Windows 10.,True **<p>** The C language insures that storage allocated with malloc is automatically garbage collected when there are no remaining references to it.,False **<p>** The C library "libc" is a portable library that can be moved from one system to another by just recompiling it with the C compiler.,False **<p>** The open file table has exactly one entry for each file open on the system.,False **<p>** The owner associated with a linux pathname will always be the same as the owner of its parent directory. For example, the owner on the file /usr1/km/share/abc, must be the same as the owner of the directory /usr1/km/share.,False **<p>** The permissions associated with a Linux pathname are stored in the directory corresponding to the last component of the pathname. For example, the permissions on the file /usr1/km/share/abc, are stored in the directory /usr/km/share.,False **<p>** The permissions associated with a Linux pathname are stored in the directory corresponding to the last component of the pathname. For example, the permissions on the file /usr1/km/share/abc, are stored in the directory /usr1/km/share.,False **<p>** The real time it takes to read frequently accessed disk blocks from a block special device is often less than with a character special device, since block special devices allows the data to be cached in the system buffer cache.,True **<p>** The real time it takes to read frequently accessed disk blocks from a block special device is often less than with a character special device, since the block special devices allows the data to be cached in the UBLOCK.,False **<p>** The SETUID bit is one bit in the "mode" of a file and determines how the EUID of a process is set when it executes the file.,True **<p>** The statement " x = x & (1<<i) " will turn on the ith bit of the integer variable if it was previously lost.,False **<p>** The statement "x=x&(1<<i)" will clear just the ith bit of the integer variable x.,False **<p>** The STDIO routine "fputc" avoids unnecessary system calls by caching output directly in the process's own U-BLOCK.,False **<p>** The STDIO routine "getc" drastically increases performance by avoiding system calls, and when implemented as a macro can even avoid the expense of a function call.,True **<p>** The Unix loader can load the UBLOCK of each process at the same address, since at run time the MMU can map this logical address to a physical address that does not conflict with processes already running.,True **<p>** Using #defined macros in a C program will often be faster than using the equivalent function, since macros avoid the heavy overhead of a context switch inherent in a TRAP instruction.,False **<p>** Using pipes to transfer data is more efficient than using message queues, since the data does not have to be copied to and from the address space of the operating system.,False **<p>** Well written Linux programs must explicitly exame arg v to process the "<" and ">" redirections.,False **<p>** When a child inherits a file descriptor from its parent they initially have the same offset, but the offsets on those file descriptors don't remain in sync with the parent,False **<p>** When a process unlinks a file and causes its link count to go from 1 to 0, its directory entry will be deleted only after all process that have the file open have exited.,False **<p>** Zombies are created when a process exits without waiting for its children.,False **<p>**