Chapter 2

UNIX

1. UNIX was developed by
- (a) Bell Labs
- (b) Berkley Software Group
- (c) California University
- (d) American Defence Academy

2. Chocolate Chip is
- (a) a latest Intel product
- (b) another name for BSD 4.2 Version
- (c) another name for System V
- (d) another name for System III

3. Which of the following features of UNIX may be used for inter process communication?
- (a) Signals
- (b) Pipes
- (c) Semaphore
- (d) Message Queues

4. Pick the incorrect statements.
- (a) Shell is a command interpreter.
- (b) Shell is the interface between user and kernel.
- (c) System can't work without a shell.
- (d) Shell is a program.

5. UNIX is
- (a) a multi-user system
- (b) a real-time system
- (c) a multi-task system
- (d) name of a file in the root directory

***6.** Which of the following statements best explains a process?
- (a) It is a program.
- (b) It is a program in execution.
- (c) It is an instance of a program in execution.
- (d) It is a program that uses system calls.

***7.** In a system, if 5 people are currently using the vi editor, then the number of corresponding processes will be
- (a) 1
- (b) 5
- (c) 2
- (d) 0

8. Kernel is not involved
- (a) when a read operation is done
- (b) when a pressed key is echoed on to the screen
- (c) in resource allocation
- (d) none of the above

***9.** The command

```
echo welcome > /dev/tty
```

- (a) echoes welcome in all the terminals that are switched on.
- (b) echoes welcome in all the terminals that are logged on.
- (c) echoes welcome only in the terminal in which it is run.
- (d) signals the error message - Terminal number not specified.

***10.** `/dev/null`
- (a) is a file
- (b) has write permission for all
- (c) is the UNIX built-in dustbin
- (d) none of the above

11. The advantage of binary files over text files is that
- (a) it is compact
- (b) it can be accessed faster
- (c) many commands (like `cat`) assume the named file to be a binary file.
- (d) they are more reliable

***12.** The permission bits of a file `noname`, can be set to `_rws_ _x_ _x` by the command.
- (a) `chmod 711 noname`
- (b) `chmod go-rw noname`
- (c) `chmod 2711 noname`
- (d) none of the above

***13.** `/bin/passwd` has the user execution permission set to `'s'` because
- (a) it is not executable
- (b) it should allow users who don't have write permission to `/etc/passwd` to write to it
- (c) `/etc/passwd is write` protected
- (d) this facility assigns to the user, permissions of the program owner, temporarily.

14. If one doesn't want anyone else to read or write to a file named `datfile`, except through a program in file `filex`, then he may use
- (a) `chmod u+s filex ; chmod go-rw datfile`
- (b) `chmod u+s datfile ; chmod go-rw filex`
- (c) `chmod 4711 datfile ; chmod go-rw filex`
- (d) `chmod 4711 filex ; chmod go-rw datfile`

15. Writing a C program that accepts input from keyboard, rather than from a file is advantageous because
- (a) keyboard is a file that is already open
- (b) it facilitates batch processing
- (c) it can be used in a pipe, if it writes to `stdout`
- (d) none of the above

***16.** Consider the following command that invokes the executable file `a.out`, with the following command line arguments `a.out God loves you`
argv[1][2] corresponds to the character

(a) `e` (b) `o` (c) `.` (d) `d`

***17.** In the previous question after the operation `argv++`, the value of `argv[1][2]` will be

(a) `e` (b) `d` (c) `v` (d) undefined

***18.** Which of the following string functions can be used to find the last occurrence of a given character in a given string?

(a) `strncmp` (b) `strncpy` (c) `strchr` (d) None of the above

19. Choose the correct statements.

(a) The function `stat` refers a file by its name.
(b) The function `stat` refers a file by its file descriptor.
(c) The function `fstat` refers a file by its file descriptor.
(d) The function `fstat` refers a file by its name.

20. Which of the following fields in the structure `stat`, has information about the permission setting of a file?

(a) `st_gid` (b) `st_mode` (c) `st_ino` (d) `st_uid`

***21.** To simulate the command `"system"`, which of the system calls - `fork`, `wait`, and `excel` is/are to be used?

(a) `fork` and `wait` (b) all three
(c) `fork` and `excel` (d) `wait` and `excel`

22. Consider the program

```
main()
{
      printf("He arose a victor from\n");
      system ("date");
      printf("the dark domain");
}
```

If a.out is the executable code corresponding to the above source code, then the command
`a.out > outf`

(a) redirects the output of date to file `outf`
(b) displays the output of date on the screen
(c) prints everything on the screen
(d) prints the two messages on the screen

23. The default permission bits of a file when it is created for the first time, is controlled by

(a) `chmod` value (b) `fmask` value (c) `umask` value (d) none of the above

***24.** Let `x.c` be a C source code. The command `cc x.c > y`

(a) is equivalent to the command `cc x.c ; mv a.out y`
(b) is equivalent to the command `cc -o y x.c`

(c) serves no purpose
(d) none of the above

25. Which of the following sections in the manual covers system calls?
(a) 1 (b) 2 (c) 3 (d) 4

26. Which of the following are not system calls?
(a) `chmod` (b) `open` (c) `lseek` (d) `getc`

27. Choose the correct statements.
(a) C programs can directly make system calls.
(b) System calls are functions used by the shell.
(c) Library functions use system calls.
(d) Library functions don't use system calls.

28. Which of the following remarks about system calls, library functions and UNIX commands are true?
(a) System call is a part of kernel, while the other two are not a part of kernel
(b) Unlike library functions, system calls and Unix commands are stand-alone programs
(c) Library functions and UNIX commands use system calls
(d) Unlike system calls, library functions and UNIX commands are stand-alone programs

29. The 2 in the manual entry `access(2)`
(a) implies `access` is a system call (b) implies `access` is a library function
(c) refers to the section number (d) none of the above

30. If path is set to `.:/usr/x:/usr/bin`, then
(a) the command one types will be first checked in the current directory, then `/usr/x` and `/usr/bin`.
(b) if a command is found in both `/usr/x` and `/usr/bin`, then the one in `/usr/x` will be executed.
(c) in the previous choice, what happens is unpredictable.
(d) if a command is found in both `/usr/x` and `/usr/bin`, then the one in `/usr/bin` will be executed.

***31.** A file `x` is created with the following contents

```
echo today is:
date
```

If you type `x`, then
(a) it echoes the message, followed by date.
(b) it gives the desired output only if the `execute` permission of file `x` is set.
(c) the desired output can be got by the command `sh x`, which works even if `x` has its `execute` permission not set.
(d) none of the above.

32. Shell script is preferable to other forms of programming because it
(a) executes faster (b) enhances portability
(c) occupies less space (d) makes programming task easier

33. Choose the incorrect statements.
(a) Shell scripts can accept arguments (b) Shell scripts are interpreted
(c) Shell is a programming language (d) Shell scripts are compiled

34. Files that store data in the same format as used in program are called
(a) binary files (b) source file (c) text file (d) core

35. To allow only one user to work with a particular file at a particular time, one has to use
(a) semaphore (b) critical region (c) locking (d) dedicated mode

36. Which of the following remarks about `realloc` are true?
(a) It allocates memory of required size that need not be contiguous
(b) It never shifts the existing block
(c) It can work only with an existing block of memory
(d) It may shift the existing block

37. The differences between `malloc()` and `calloc()` are:
(a) `malloc` is used for dynamic allocation of memory, while `calloc` can't be used for that purpose.
(b) `malloc` needs only one argument, while `calloc` needs two.
(c) unlike `malloc`, `calloc` allocates memory and initializes it to `0`.
(d) `malloc` needs two arguments and `calloc` only one.

38. The file that stores an integer as a sequence of characters is a
(a) text file (b) data file (c) binary file (d) core

39. If `cat x`, prints garbage, then `x` is probably a
(a) data file (b) binary file (c) text file (d) source file

***40.** Which of the following file names can be found in more than one directory?
(a) `passwd` (b) `bin` (c) date (d) none of the above

41. `/bin`
(a) is a bucket for storing information (b) has files in binary code
(c) is a directory (d) none of the above

42. The main reasons for the success of pipes are
(a) the availability of many filter programs
(b) UNIX treats devices as files
(c) it provides a 2-way communication channel
(d) all of the above

43. Which of the following are not filter programs?
(a) `date` (b) `sort` (c) `cat` (d) `grep`

44. Redirection in pipes can be achieved by using

(a) `>` (b) `>>` (c) `tee` (d) `lpr`

45. Choose the correct statements.

(a) The symbols > and | are both processed by shell

(b) > can be used to direct output to a named file

(c) | can be used to direct output to programs

(d) Filter programs can be piped

***46.** The command `who | sort - file1 > file2`

(a) results in an error

(b) sorts the contents of `file1` and puts it in `file2`

(c) puts in `file2`, the sorted output of who, followed by sorted contents of `file1`

(d) none of the above

***47.** If the command `cat x`, is executed after successfully executing the command `time sort filename > x`, then

(a) only the time details will be displayed

(b) only the sorted contents of the file `filename` will be displayed

(c) an error message will be displayed

(d) the sorted contents of the file `filename`, along with the `time` information will be displayed

48. Which of the following information is not present in an i-node?

(a) Contents of the file (b) Size of the file

(c) Name of the file (d) Permission setting of the file

49. The system identifies a file by its

(a) name (b) absolute path (c) file owner (d) inode number

50. The system identifies the end of a file by the

(a) EOF character (b) file size (c) i-node number (d) none of the above

***51.** The command line argument `a.out x 'a b' "c d"`

(a) is acceptable

(b) is acceptable if the double quotes are replaced by single quotes

(c) is acceptable if the single quotes are replaced by double quotes

(d) none of the above

52. Which of the following metacharacters will be recognized by the shell, even if it comes within double quotes?

(a) `$` (b) `*` (c) `?` (d) None of these

***53.** `lint` should be used

(a) before compilation (b) after compilation

(c) to analyze a C code (d) none of the above

54. Environment variables can be accessed by

(a) system programs (b) C programs
(c) shell scripts (d) none of the above

55. Which of the following are character special files?

(a) Terminal (b) Printer (c) Modem (d) Tape Drive

56. If one exports a variable

(a) variables placed in the environment by a child process are not inherited by the parent process.
(b) it is passed to all its descendant processes
(c) it dies when the shell that created it dies
(d) only the first two choices are correct

57. Profilers are

(a) tools that analyze the run time behaviour of a program
(b) tools that check a C code for cross file consistency
(c) tools that keep track of evolving versions of a file
(d) none of the above

58. The shell command :

(a) does nothing
(b) can be used to cause infinite looping
(c) can take arguments but it cannot act on them
(d) can be used to indicate a comment

***59.** Which of the following tools can be used to keep track of evolving versions of a file?

(a) `make` (b) `yacc` (c) `sccs` (d) `dv`

***60.** The . (dot) shell command

(a) can take command line argument
(b) will fork a child shell to execute the named shell script
(c) can be used to change the environment of the current shell
(d) all of the above

***61.** `m4`

(a) is a macro processor
(b) can be used to preprocess C code
(c) can be used to preprocess assembly language program
(d) none of the above

***62.** The first thing that is searched when a command references a file is its

(a) i-node (b) i-node number
(c) permission setting (d) none of the above

63. `cc` command sequentially invokes
(a) preprocessor, compiler and link editor
(b) compiler and link editor
(c) preprocessor, compiler, assembler and link editor
(d) compiler, assembler and link editor

***64.** Among the directory entries, i-node and the file contents, which will be changed when a file is updated?
(a) Only directory entry and file contents (b) Only i-node and file contents
(c) All the three (d) None of the above

65. The `cc` command
(a) can take more than one argument
(b) can act on files with `.c` or `.o` extension
(c) creates `.o` files by default when more than one argument with `.c` extension is present
(d) if provided with more than one argument, immediately terminates if the first argument fails to compile successfully

***66.** The `mv` command changes
(a) only the directory entry (b) only the directory entry and i-node
(c) only the i-node number (d) none of the above

67. If 7 terminals are currently logged on, then the command
`date ; who | wc -l`, displays
(a) `date` followed by `7` (b) `date` followed by `8`
(c) `date` followed by `1` (d) an error message

***68.** Choose the correct answers if the command `ls -l /dev/mt0` displays
```
brw_rw_ _ _ _ 1 root 3, 0 jan 18 11:05 mt0
```
(a) The `'b'` indicates that it is a special file
(b) `mt0` indicates that it is a tape drive
(c) `mt0` indicates that it is a mounted tape
(d) The `'b'` indicates that data transfer is done in blocks

69. Choose the correct statements.
(a) `ld x.o` is a valid command (assume `x.o` exists)
(b) `ld x.o` is same as `cc x.o`
(c) `cc x.s` is a valid command (assume `x.s` exists)
(d) All of the above

***70.** `cat /dev/tty`
(a) throws garbage onto the terminal `tty`
(b) just echoes what you type, line by line
(c) terminates if one types control `d`, at the beginning of a line
(d) terminates if one types control `d`, anywhere in a line

71. The header files used in C programs are usually found in

(a) `/bin/include` (b) `/usr/bin/include`

(c) `/dev/include` (d) `/usr/include`

72. The command `pwd` displays `/x/y`. After executing the command `chmod u-x`, which of the following commands will not work?

(a) `cd ..` (b) `ls` (c) `chmod u+x` (d) `pwd`

73. A C program should be compiled with `-g` option (like `cc -g x. c`) to use

(a) `prof` (b) `make` (c) `lprof` (d) `sdb`

74. The difference between a pipe and a regular file is that

(a) unlike a regular file, pipe is not a file.

(b) the data in a pipe is transient, unlike the contents of a regular file.

(c) pipes forbid random accessing, while regular files do allow this.

(d) all of the above.

75. Choose the correct statements.

(a) The default linking arrangement for `cc` is dynamic.

(b) Dynamically linked programs save disk storage.

(c) Dynamically linked programs enhances shareability of library routines.

(d) Dynamically linked programs can be fixed or enhanced without relinking the applications that depend on it.

76. Context switch changes the process mode from

(a) user to kernel mode

(b) kernel to user mode

(c) kernel mode to the kernel process

(d) kernel process to the kernel mode of some process

77. File `x.c` has 5 lines of code. The command

`date | tee abc | sort - x.c | wc -l`, displays

(a) `5` (b) `6` (c) `0` (d) an error message

78. Which of the following comments about the signals system call are true?

(a) It takes up two arguments

(b) The second argument, is a function call

(c) The second argument is a pointer to a function

(d) The first argument is an integer

79. `lint` can analyze the named source code for

(a) inconsistent usage (b) non portability

(c) suspicious constructs (d) none of the above

80. Which of the following characteristics of the original process are preserved when, the `exec` system call is executed?

(a) The current working directory (b) The open files

(c) PID (d) PPID

*81. Which of the following remarks about lex are true?

(a) It generates a C program.

(b) It produces a C code that consumes more memory than a C program that can be written separately to accomplish the same task.

(c) It produces a C code that executes slower than a C program that can be written separately to accomplish the same task.

(d) None of the above.

82. Which of the following programs are not interactive?

(a) `passwd` (b) `date` (c) `grep` (d) `sh`

83. `lex` can be used for

(a) text processing

(b) code enciphering

(c) compiler construction

(d) collecting statistical data of different patterns

*84. The number of errors in the following shell script

```
echo How are you ?
read $answer
```

is

(a) 0 (b) 1 (c) 2 (d) 3

85. The `read` in the previous question is a

(a) library function (b) system call (c) shell command (d) none of the above

86. If `lex.l` is a `lex` code then

(a) the command `lex lex.l` invokes `lex` to act on `lex.l`

(b) the command `lex lex.l` writes its output to the file `lex.yy.c`

(c) `lex.yy.c` has the definition of the function `yylex`

(d) `lex` library can be invoked by the compiler option `ll`

87. Choose the correct statements.

(a) Any process has an associated owner ID and group ID.

(b) Effective ID defines who you are for the duration of a process.

(c) Real ID defines who you are for the duration of a process.

(d) Effective ID is available in `/etc/passwd` file.

*88. A file `hai` has the following shell script in it

```
echo Oh! What a wonderful day
echo Day I will never forget 1>&2
echo Day I will never ever get
```

The command `sh hai > mn`

(a) puts all the three messages in mn

(b) puts the second message both in mn and the screen

(c) puts only the first and the third message in `mn`

(d) results in an error

***89.** No shell script can take input from

(a) `stdin`

(b) the output of the previously executed command redirected to it

(c) the file that holds the script

(d) none of the above

***90.** The command `cc x.c && a.out`

(a) is equivalent to `cc x.c ; a.out`

(b) means execute `a.out` only when `x.c` compiles successfully

(c) means execute `a.out` only if `cc x.c` returns a value `0` to the system

(d) all of the above

91. Which of the following shell script's looping features does not recognize the `break` command?

(a) `while` (b) `until` (c) `for` (d) None of the above

92. Shell script

(a) needs no compilation

(b) is ideal for manipulating a file, character by character

(c) is not good in arithmetic operations

(d) enhances portability

93. The desirable features of a new shell script you write is that

(a) it should take its input from `stdin`

(b) on successful termination, it should exit with a non-zero value

(c) it should not accept command line arguments

(d) it does some cleaning up operation, on termination

94. Which of the following shell commands displays the contents of each of the command line arguments, one by one?

(a) `cat $*` (b) `cat '$*'` (c) `cat "$@"` (d) `cat "$*"`

95. The disadvantage of a pipe is that

(a) it is a one way communication channel

(b) it dies along with the process that created it

(c) it can't be shared by unrelated processes

(d) none of the above

96. The state of signals are

(a) preserved across a `fork` call

(b) not preserved across a `fork` call

(c) not preserved across an `exec` call

(d) preserved across an `exec` call

97. A `fork` system call will fail, if

(a) the previously executed statement is also a `fork` call.

(b) the limit on the maximum number of processes in the system would be exceeded.

(c) the limit on the maximum number of processes that can be under execution by a single user would be exceeded.

(d) all of the above.

98. Which of the following options for the shell command `test` should be followed by the file descriptor?

(a) `r` (b) `d` (c) `t` (d) `s`

99. Which of the following displays the `exit` status of the last executed command?

(a) `echo $#` (b) `echo $$` (c) `echo $?` (d) `echo $!`

***100.** Which of the following file names cannot be displayed if `ls *` is run?

(a) `-Xy` (b) `?x` (c) `.x` (d) `hidden`

101. Which of the following initiates the sequence of events that ultimately allows a user to `login`?

(a) `clri` (b) `sync` (c) `login` (d) `init`

***102.** `getc(stdin)`

(a) results in run time error (b) results in syntax error

(c) is equivalent to `getchar()` ; (d) none of the above

103. Which of the following is not the work of a C-preprocessor?

(a) Macro expansion (b) File inclusion

(c) Conditional compilation (d) None of the above

104. Which of the following is used to write disk block images from memory to disk?

(a) `clri` (b) `sync` (c) `mkfs` (d) `stty`

***105.** Choose the correct statement.

(a) To read successive characters from an open file, `getchar` and `scanf` can be used interchangeably.

(b) To read successive characters from an open file, `getchar` and `read` can be used interchangeably.

(c) The `read` system call reads from the buffer.

(d) None of the above.

***106.** The following program

```
main()
{
     close(1);
     print("How R U?");
}
```

(a) is syntactically incorrect (b) results in a run-time error

(c) will wait indefinitely, if executed (d) none of the above

107. The PID of the kernel process is
(a) undefined (b) `0` (c) `1` (d) `3`

***108.** Choose the correct remarks.
(a) `exit` and `return` can be used interchangeably
(b) Use of `return` terminates the program
(c) Use of `exit` terminates the program
(d) `exit` returns a value to the system

109. Which of the following is an index to the array of open files maintained by the kernel for a user?
(a) i-node (b) i-node number (c) File descriptor (d) File pointer

110. In which of the following directories does `init` reside?
(a) `root` (b) `bin` (c) `etc` (d) `usr`

111. The command `cat > x`
(a) is invalid
(b) creates a file x and displays an error message
(c) creates a file x and waits for the user to give input from the keyboard
(d) none of the above

112. Which of the following are defined in `stdio.h`?
(a) EOF (b) NULL (c) BUFSIZE (d) None of the above

113. The `login prompt` can be changed by changing the contents of the file
(a) `inittab` (b) `init` (c) `passwd` (d) `gettydefs`

114. When the read system call encounters EOF, it returns
(a) some positive integer (b) some negative integer
(c) `0` (d) `-1`

115. Which of the following library functions do not `return` a pointer to the structure FILE?
(a) `fopen` (b) `fclose` (c) `freopen` (d) `fwrite`

***116.** Which of the following processes are involved in the process of allowing a person to `login`?
(a) `init` (b) `getty` (c) `login` (d) kernel

***117.** Which of the following system calls reads 8 bits from the standard input? (assume `buff` is a pointer to the buffer area)
(a) `read(0, buff, 8)` (b) `read(1, buff, 8)`
(c) `read(0, buff, 1)` (d) `read(1, buff, 1)`

118. Which of the following are implemented as macros (rather than functions)?
(a) `getchar` (b) `getc` (c) `fgetc` (d) `fputc`

119. Choose the correct statements.
(a) `errno` is an external variable available to any 'C' program
(b) `errno` is set to a value when an error occurs
(c) `errno` is cleared when a nonerroneous call is made
(d) `errno` cannot be used to find the cause of an error

120. When the user responds to `login` prompt
(a) `getty` forks `login` process (b) `login` process replaces `getty` process
(c) a shell will be created (e) none of the above

121. The shell command `cat x y > x`
(a) doesn't work
(b) replaces the contents of file `x`, by the contents of file `y`
(c) does nothing, other than displaying an error message
(d) none of the above

122. Which of the following return file descriptor?
(a) `close` (b) `fopen` (c) `open` (d) `creat`

123. To simulate the `who` command, one has to access the file
(a) `/etc/passwd` (b) `/bin/.login`
(c) `/etc/utmp` (d) `/usr/user_dat`

124. A file system in UNIX has the four sections—boot block, super block, I-list and data block that are arranged in the order
(a) boot block, super block, I-list and data block
(b) boot block, data block, super block and I-list
(c) boot block, data block, I-list and super block
(d) super block, boot block, data block and I-list

125. `stderr, stdout, stdin` have the file descriptors
(a) 0, 1, 2 respectively (b) 0, 2, 1 respectively
(c) 1, 0, 2 respectively (d) 2, 1, 0 respectively

126. Which of the following functions can be used to randomly access a file?
(a) `fgetc` (b) `getc` (c) `fseek` (d) `ftell`

127. A manual entry of the form `xyz(3S)`
(a) implies `xyz` is a system call
(b) implies `xyz` is a library function
(c) means `xyz` is a library function that is part of the standard `i/o` package
(d) means `xyz` is a library function that is a part of the standard math library

128. The reference time adopted by UNIX is
(a) Jan 1, 1970 (b) Jan 1, 1980 (c) Jan 1, 1982 (d) Jan 1, 1972

***129.** `perror()` can be simulated by using
(a) `errno` and `sys_nerr` (b) `sys_errlist` and `sys_nerr`
(c) `sys_errlist` and `errno` (d) none of the above

130. A process that uses CPU, cannot continue to use it if
(a) the CPU time slice expires (b) a higher priority process arrives
(c) it has to wait for an event to happen (d) it executes an `exit` statement

131. The command `cd ./../.`

(a) serves no purpose (b) is invalid

(c) is equivalent to `cd ..` (d) none of the above

132. The UNIX tool `awk`

(a) can do both numerical and string comparison

(b) decides from the context whether the comparison is numerical or alphabetical

(c) signals an error if an alphabet is compared with a number

(d) all of the above

***133.** Which of the following strings will be matched by `awk`, if `/(x +)*y!$/` is the specified pattern to be searched for?

(a) `x +x +x +y!$` (b) `x x x x y!$`

(c) `x x x xy!` (d) none of the above

134. When `awk` encounters strings in arithmetic expressions,

(a) it treats them as having the value `0` (b) it treats them as having the value `1`

(c) it displays an error message (d) it is assigned an arbitrary value

135. Which of the following comments about `awk` are true?

(a) It is a text processing language (b) Arrays can be indexed by string

(c) It has features for redirecting its output (d) None of the above

***136.** Which of the following UNIX tools, receives input only from the standard input?

(a) `awk` (b) `grep` (c) `sed` (d) `tr`

137. If `x.c` is a file, then `ed x.c` creates a copy of `x.c` in

(a) `/etc` (b) `/usr` (c) `/tmp` (d) `/usr/bin`

***138.** The number of 3's in the output of the following C program

```
main()
{
   printf("1"); fork();
   printf("2"); fork();
   fork(); printf("3");
}
```

is

(a) 1 (b) 8 (c) 4 (d) 2

139. Which of the following processes has the PID 1?

(a) kernel (b) unix (c) init (d) shell

140. Which of the following remarks about `fgrep` are true?

(a) It is faster than `grep`.

(b) It is compact to use.

(c) It does not recognize any meta-character.

(d) It can simultaneously search for different patterns.

***141.** Which of the following results in an error?

(a) `expr 4+5` (b) `expr 9-3` (c) `expr 2*3` (d) `expr 7/5`

142. Which of the following is not a command delimiter?

(a) `new line` (b) `;` (c) `&` (d) `,`

143. A file `abc` has the following shell script in it.

```
cat $1 > $1.$$
```

The command `sh abc file1`

(a) results in an error

(b) is equivalent to `cp $1 $1.$$`

(c) copies the contents of `file1` to another file that has the PID of the executing shell as its extension

(d) none of the above

144. `*?*` will be the output of

(a) `echo *?*` (b) `echo '*?*'` (c) `echo "*?*"` (d) `echo \*\?\*`

145. Which of the following shell variables can be used to customize the editors (like `ex`, `vi`)?

(a) `PATH` (b) `IFS` (c) `HOME` (d) `EXINIT`

Go through the following sequence of commands and answer the next two questions based on it.

```
$echo $x
$sh
$x=hai
$export x
$sh
```

***146.** `echo $x` will output

(a) `hai` (b) garbage

(c) an empty line (d) none of the above

***147.** If the command exit is run twice followed by running the command `echo $x`, the output will be

(a) `hai` (b) garbage

(c) an empty line (d) none of the above

***148.** An orphan process

(a) is a child process that was terminated before the parent process

(b) is adopted by the login shell

(c) is adopted by the process dispatcher

(d) will be denoted by the process status O

***149.** Which of the following calls never returns an error?

(a) `getpid` (b) `fork` (c) `ioctl` (d) `open`

***150.** The following C program

```
main()
{
   fork(); fork(); printf("yes");
}
```

prints `yes`

(a) only once (b) twice (c) 4 times (d) 8 times

***151.** When a process makes a system call, its mode changes from

(a) user to kernel

(b) kernel to user

(c) restricted to unrestricted

(d) unrestricted to restricted

***152.** Choose the correct statements.

(a) When a process makes a system call, a `context` switch is initiated.

(b) Kernel is not involved in servicing a system call.

(c) When a process making a system call has to wait for an event to occur, then a process switch to the kernel process is initiated.

(d) System calls cannot be serviced in kernel mode.

***153.** The command `ls > xy`

(a) displays an error message, if `xy` exists and is `write` protected

(b) if followed by `cat xy`, lists `xy` also

(c) redirects errors, if any, to `xy`

(d) none of the above

154. Shell functions

(a) are another name for shell procedures (b) execute faster than shell procedures

(c) are executed by a new shell (d) are not executed by a new shell

***155.** The `cc` command makes a total of

(a) 1 pass (b) 2 passes (c) 4 passes (d) 5 passes

156. Which of the following is not invoked when the `cc` command executes?

(a) `/lib/cpp` (b) `/lib/cl` (c) `/bin/as` (d) `/bin/ld`

157. `creat` will fail, if

(a) there are too many open files

(b) the `filename` is a directory

(c) the named file already exists with its `write` permission off

(d) the parent directory of the named file is `write` protected

158. Which of the following arguments to the open system call, will be discarded, if the named file already exists?

(a) `O_TRUNC` (b) `O_APPEND` (c) `O_EXCL` (d) `O_CREAT`

159. Under which of the following circumstances `rm /y/x`, cannot remove `x`?

(a) If `x` is `write` protected, but `y` is not write protected.

(b) If `x` is not write protected, but `y` is `write` protected.

(c) If `y` has its execution permission bit off.

(d) All of the above.

160. File pointer

(a) is a `long` integer

(b) is of pointer data type

(c) represents the position of the `read-write` head from the beginning of the file.

(d) none of the above

***161.** The C compiler can be modified to compile programs coded in other high level languages just by changing

(a) `/lib/ccom` (b) `/lib/c2` (c) `/lib/c1` (d) `/bin/as`

162. When a file is aliased

(a) a new directory entry is created (b) a new i-node is created

(c) the i-node number is shared (d) none of the above

163. Setting the execute bit on has no meaning, if the file is a

(a) directory (b) shell script (c) C source code (d) symbol table

164. Which of the following sections of an executable binary file has all uninitialised data items?

(a) `bss` (b) Data (c) Header (d) Symbol table

165. In which section of a process, the information about the arguments to the program are available?

(a) Data (b) Text (c) Stack (d) User-block

166. Which of the following system calls transforms an executable binary file into a process?

(a) `fork` (b) `exec` (c) `ioctl` (d) `longjmp`

167. UNIX was first installed in

(a) IBM-360 (b) PDP/11 (c) PDP/7 (d) CRAY

168. PID is used by the system to identify

(a) a process (b) the file name (c) the i-node (d) all of the above

169. Choose the best answer.

Suspended processes are written onto a

(a) swap area (b) dedicated area (c) ROM (d) critical area

***170.** Which of the following system calls, does not return control to the calling point, on termination?

(a) `fork` (b) `exec` (c) `ioctl` (d) `longjmp`

***171.** Which of the following remarks about the `return` value of "wait" are true?

(a) In case of normal termination (through `exit`), the lower byte of the `wait` status is set to zeroes

(b) In case of abnormal termination, the lower byte of the `wait` status is set to zeroes

(c) A core dump sets the seventh bit on

(d) A process in `zombie` status sets the seventh bit on

***172.** The following C program

```
main()
{
   printf("WHATIZIT");
   system("date");
}
```

(a) first prints `WHATIZIT` and then displays the output of `date` command in the next line.

(b) first prints `WHATIZIT` and then displays the output of `date` command in the same line.

(c) first displays the output of `date` command and then `WHATIZIT` in the next line.

(d) none of the above.

***173.** The program

```
main()
{
     printf("x");
     fflush(stdout);
     system("date");
}
```

(a) gives the same output as the program

```
main()
{
     printf("x\n");
     system("date");
}
```

(b) prints `x`, before displaying date

(c) prints `x` after displaying date

(d) all of the above

***174.** An attempt to read from a locked file, results in

(a) prematured termination (b) a deadlock

(c) an indefinite wait (d) none of the above

***175.** Which of the following is not a valid argument to the function `main` in a C program?

(a) `errno` (b) `argc` (c) `envp` (d) `argv`

176. Mounting a file system results in the loading of

(a) boot block (b) super block (c) i-node table (d) all of these

***177.** Choose the correct statements.

(a) If two users execute a file, two copies will be there in memory

(b) Shareable programs are loaded into swap area

(c) `chmod u+t filename`, is a valid command

(d) None of the above

***178.** Go through the following C program

```
main()
{
     int i, n;
     for(i = 1 ; i <= n ; ++i)
     fork();
     printf("yes");
}
```

For what value of `n`, will `yes` be printed 24 times?

(a) 3 (b) 4

(c) 5 (d) Impossible to find such an `n`

***179.** Consider the following program

```
main()
{
     printf("God looks at the heart, not the hand\n");
     system("date");
     printf("The giver, not the gift");
}
```

If `a.out` is the executable file corresponding to the above program, then the command
`a.out > x ; cat x`

(a) displays both the messages, with the output of `date` coming in between

(b) displays the output of `date` before both the messages

(c) does not display the first message

(d) none of the above

***180.** The following program

```
main()
{
      if(fork() > 0)
      sleep(100);
}
```

results in the creation of

(a) an orphan process (b) a `zombie` process

(c) a process that executes for ever (d) none of the above

181. In UNIX, the status of a process may be
(a) `running` (b) `orphan` (c) `sleeping` (d) `zombie`

182. Consider the following program

```
main()
{
    int i = 7;
    if(0 == fork())
    i += 10;
    else
    {
        wait(0);
        printf("%d", i);
    }
}
```

Choose the correct answers.

(a) The statement `i += 10` is executed by the child only

(b) The statement `i += 10` is executed by the parent only

(c) The child can start executing, only after the termination of the parent process.

(d) None of the above

***183.** The value of `i`, printed by the above program will be
(a) `10` (b) `7` (c) `17` (d) none of the above

***184.** The exception to the fact that any process in UNIX, has a parent is
(a) `dev` (b) `sh` (c) kernel (d) `login`

185. Which of the following are shared between a parent process and a child process?
(a) External variables (b) Pointer variables
(c) File pointers (d) Pipes

***186.** Consider the following C program

```
main()
{
    int j = 7, *i = &j;
    if(0 == fork())
    *i = (*i + 10);
    else
    {
        wait(0);
        printf("%d", *i);
    }
}
```

The value of `i` that will be printed is

(a) 10 (b) 7 (c) 17 (d) none of the above

***187.** In the previous question, if the declarations are made global (i.e., declared before `main()`), then the value of `i` that is printed will be

(a) 10 (b) 7 (c) 17 (d) none of the above

188. Choose the correct statements.

(a) Interrupts are caused by events that are external to a process.
(b) An exception condition is caused by an event external to a process.
(c) An exception condition happens in the middle of the execution of an instruction.
(d) An interrupt happens in the middle of the execution of an instruction.

***189.** Consider the following program

```
#include<signal.h>
mn()  ;
main()
{
        signal(SIGINT, mn);
        for (; ;)    ;
}
mn()
{
        printf("x\n");
}
```

On receipt of the signal `SIGINT`

(a) the default action corresponding to `SIGINT` will be performed
(b) the user defined function `mn`, will be executed
(c) what happens depends on whether the signal is received for the first time or not
(d) none of the above

***190.** In the previous question, if the statement
`signal(SIGINT, mn);` is repeated thrice, then

(a) what happens depends on whether the signal is received for the first time or not.
(b) what happens depends on whether the signal is received for the fourth time or not.
(c) it cannot print the message more than three times
(d) none of the above

***191.** Which of the following comments about semaphore are true?

(a) It is an integer that can act as a counter
(b) Its value depends on the number of resources to be shared
(c) Its value is stored in the kernel
(d) It can be used for resource synchronization

***192.** The following sequence of commands

```
grep x *.c > mn&
wc -l mn&
rm mn&
```

produces the same result as the single command

(a) `grep x *.c | wc -l`

(b) `wc -l < grep x *.c`

(c) `grep x *.c > wc -l`

(d) none of the above

193. Choose the correct statements.

(a) Kernel is non-preemptive.

(b) Interrupts are blocked when critical section of a code is being executed.

(c) No process can put another process to sleep.

(d) None of the above.

194. Choose the correct statements.

(a) A disk cannot have more than one file system stored in it.

(b) On the logical level, the kernel deals with disks rather than file system.

(c) The logical to physical device address mapping is done by the device driver.

(d) None of the above.

195. Which of the following data structures is not maintained by the kernel?

(a) User file descriptor table (b) File table

(c) I-node table (d) None of the above

196. Choose the correct statements.

(a) A file has only one associated i-node.

(b) I-node stands for index node.

(c) A particular i-node may correspond to more than one file.

(d) A file can have more than one associated i-node.

197. The call `pipe(p);` is valid if `p` had been declared as

(a) `int p` (b) `int p[2]` (c) `char *p` (d) `FILE *p`

198. Choose the correct statements.

(a) When a program terminates, pipes are automatically closed.

(b) If the `write` end of a pipe is closed then an attempted `read` from the other end results in a deadlock.

(c) If the `write` end of a pipe is closed, then an attempted `read` from the other end, terminates the program.

(d) None of the above.

***199.** Consider the following program

```
#include<signal.h>
main()
{
        signal(SIGINT, mn);
        fork();
        fork();
        for(; ;);
}
mn()
{
        printf("x\n");
}
```

Choose the correct statemets.

Pressing the `<del>` key

(a) sends the signal, only to the parent process

(b) sends the signal, to all the four processes

(c) for the first time, prints x only once

(d) for the first time, prints x four times

***200.** Consider the following program

```
main()
{
  int p[2]
  pipe (p);
  fork();
}
```

Choose the correct statements.

(a) The pipe will be recognized only by the parent process.

(b) `p[0]` is the file descriptor of the `write` end of the pipe.

(c) There will be four file descriptors in memory.

(d) The pipe will be shared by both the parent and the child processes.

Answers

1. a	2. b	3. a, b, c, d	4. c	5. a, c, d
6. c	7. b	8. d	9. c	10. a, b, c
11. a, b, d	12. d	13. b, c, d	14. a, d	15. a, c
16. d	17. c	18. d	19. a, c	20. b

21. b	22. a	23. c	24. c	25. b
26. d	27. a, c	28. a, c	29. a, c	30. a, b
31. b, c	32. b, c, d	33. d	34. a	35. c
36. c, d	37. b, c	38. a	39. b	40. a, b
41. b, c	42. a, b	43. a	44. c	45. a, b, c, d
46. d	47. b	48. a, c	49. d	50. b
51. a	52. a	53. a, c	54. a, b, c	55. a, b, c
56. a, b, c	57. a	58. a, b, c, d	59. c	60. c
61. a, b, c	62. b	63. c	64. b	65. a, b, c
66. a	67. a	68. a, b, d	69. a, c	70. b, c
71. d	72. a, b, c, d	73. d	74. b, c	75. a, b, c, d
76. a, b	77. b	78. a, c, d	79. a, b, c	80. a, b, c, d
81. a, b, c	82. b, c	83. a, b, c, d	84. c	85. c
86. a, b, c, d	87. a, b	88. c	89. d	90. b, c
91. d	92. a, c, d	93. a, d	94. a, c	95. a, b, c
96. a, c	97. b, c	98. c	99. c	100. b, c
101. d	102. c	103. d	104. b	105. a
106. d	107. b	108. c, d	109. c	110. c
111. c	112. a, b, c	113. d	114. c	115. b, d
116. a, b, c, d	117. c	118. a, b	119. a, b	120. b
121. b	122. c, d	123. c	124. a	125. d
126. c, d	127. b, c	128. a	129. c	130. a, b, c, d
131. c	132. a, b	133. d	134. a	135. a, b, c
136. d	137. c	138. b	139. c	140. b, c, d
141. c	142. d	143. b, c	144. b, c, d	145. d
146. a	147. c	148. a, c, d	149. a	150. c
151. a, c	152. a, b, c	153. a, b	154. b, d	155. d
156. b	157. a, b, c, d	158. d	159. b, c	160. a, c
161. a	162. a, c	163. c	164. a	165. c
166. b	167. c	168. a	169. a	170. b
171. a, c, d	172. c	173. b	174. d	175. a
176. b, c	177. a, b, c	178. d	179. b	180. b
181. a, b, c, d	182. a	183. b	184. c	185. c, d
186. b	187. b	188. a, c	189. c	190. a
191. a, b, c, d	192. d	193. a, b, c	194. c	195. d
196. a, b, c	197. b	198. a, c	199. b, d	200. c, d

Explanations

6. Process is a program in execution. However, if many users are simultaneously executing the same program, the system has to differentiate the instances of the program, in use. Hence the best answer is (c).

7. Refer question 6.

9. `/dev/tty` is a synonym for the terminal you are currently using. If `echo welcome > /dev/tty`, is a part of a shell, `welcome` will be echoed in the terminal in which the script is run, doesn't matter which terminal it is.

10. `/dev/null` can be called UNIX built-in dust-bin. To prevent a program from filling the monitor with garbage, `/dev/null` comes in handy. Just redirect it to `/dev/null`. It gladly accepts garbage. It is a universal sink.

12. We can use the command `chmod 711 noname`, followed by `chmod u+s noname` (use `ls -l noname and check`). Else use the single command `chmod 4711 noname`. What is this "s" anyway? Only the super user has the permission to change `/etc/passwd file`. But any user can update it through the `passwd (/bin/passwd)` command. If you type `ls -l /bin/passwd` you can see the user execution bit set to `s` instead of x). It is because of this "`s`", a user can access `/etc/passwd` through the passwd command, for which he is not otherwise entitled to.

13. Refer to Question 12.

16. `argv[0]` is a pointer to the command (here `a.out`) and `argv[1]` to the first argument. So, `argv[1][2]`, refers to the third (since count begins at `0`) character of the first argument which is `d`.

17. `argv++`, makes `argv[0]` point to the first argument. So, after `argv++`, `argv[0]` will be pointing to God and `arg[1]` to `loves`. So, `argv[1][2]` will be `v`.

18. `strrchr()` is the correct function. It returns a pointer to the last occurrence of the character specified as argument.

21. When a process executes a system command like system("date"), it forks a child process, waits for it to execute the `date` command and terminates. This can be implemented by the child making the call

```
execl("/bin/date",   "date", (chr *)0)
```

24. The redirection symbol >, puts everything that will otherwise be displayed in the screen, to the named file (`y` here). If `x.c` is syntactically correct, then the command `cc x.c`, silently creates `a.out`, but what comes to the screen is nothing (other than the next prompt). So, `y` will be empty.

31. When you create a file using an editor, it will be assigned default permission setting (determined by the umask value). Generally the execute permission will be off. So, to run a shell script, set its execute bit on. However, if you run `sh x`, `x` will be executed, even if its execute bit is off.

40. `passwd - /etc/passwd and /bin/passwd`

`bin - usr/bin and /bin`

46. The sort combines the output of `who` (`'-'` stands for output of the previous command) and contents of `file1`, and sorts the resultant. So, `who | sort - file1 > file2` is equivalent to the sequence of commands `who > x, cat x file1 > y, sort y > file2`. Hence the correct answer is (d).

47. The time command uses `stderr`, instead of `stdout` to display its results. As a result of this, what is redirected to `x` is just the output of `sort filename` command and not the time details. The time details will be displayed in the screen, since screen by default is the `stderr`.

51. Any command that is keyed-in will be first processed by the shell, which divides the command into tokens, taking into account the metacharacters. So, `a b` will be divided into two tokens `a` and `b`. But `'a b'` or `"a b"` will pass `a b` as such (i.e. as one token). So, `argv[2]` will be `'a b'` and `argv[3]`, `"c d"`. Hence the answer.

53. `lint` can throw light on many things, which the compiler generally overlooks. So, potential errors can be spotted and the program debugged, even before compilation. Hence the answer is (a) & (c).

59. `SCCS` stands for Source Code Control System. Many applications need periodical updation of files (e.g., master file in a business application). It is always better to have backup of the changed version, for security reasons and undo operations. `SCCS` is a UNIX tool that can be used to keep history of the changes made.

60. Any shell script will not be executed by the current shell. The current shell forks a new shell that executes the named shell script and terminates after it. So, any variable exported in the shell script will not be recognized by the parent shell. The `.` (dot) command makes the named shell script to be executed by the current shell. On the negative side, `.` (dot) commands (like, profile) don't accept command line arguments.

61. `cpp` is 'C' preprocessor, `m4` is a general purpose macro processor that can be used to preprocess C, as well as assembly language programs. Unlike `cpp`, it can do integer arithmetic, some string and substring manipulation, in addition to file inclusion and conditional macro expansion which can be done by `cpp` also.

62. Suppose you enter a command like `cp x y`. Unlike the user, who uses the name to identify and differentiate files, the system uses i-node number to uniquely identify a file. Any file name has an associated i-node number. In UNIX, different files can have the same name. But the associated i-node number will be different. The filename—i-node correspondence can be found in the directory which has to be the first one that is to be searched, as nothing can be done to a file without knowing its i-node number.

64. Directory entries have two fields. One for the file name and the other for the i-node number. The i-node has many fields for storing all the information about the file, except the file name and the actual content of the file. The content of the file will be in a separate place. So, the details of any file will be spread over these three places. When a file is updated its name and i-node number will remain the same. Only the contents and some fields in the i-node (like file size, time of last access, etc.) need to be changed. Hence the answer.

66. The `mv` command, say, `mv x y` is not going to change the file content, the i-node number or other information in the i-node. Only the file name is going to change. The file name is present only in the directory. So, the answer is (a).

68. For regular (ordinary) files the first character (i.e. b here), will be just `a underscore`. For directories `d`, for character special files `'c'` and `'b'` for a block read special file. The last column will have `lp` for line printer, `hp` for disk drives, `tty` for terminals etc. The `3` in `3, 0` denotes the major device number and `0`- minor device number. That is, this system denotes tape drives by `3` and `0` to single out a particular tape drive from the many tape drives, the system may have.

70. First, the i-node number corresponding to `/dev/tty` (i.e. the terminal currently used) is procured. Then the i-node is accessed. From it, the system understands, it is a character special file. So, whatever you type, if followed by `'\n'` will be echoed in the terminal. Typing control `d`, also flushes the buffer contents to `tty`. But unlike `'\n'`, control `d` is not transmitted. So, if you type `ab(^d)cd(^d)` first `ab` will be immediately transmitted, then `cd` will be transmitted. Whenever you press control `d`, then what you have typed between the previous control `d` (or from the start of the current line) to the current control `d` will be transmitted. So, if you type two control `d` consecutively or a single control `d`, at the beginning of a line then you are telling it to transmit, but nothing is there to be transmitted. So, the command gets terminated.

81. The purpose of lex is to generate a 'C' function `yylex`, that will recognize any pattern that is given as input to lex, as a regular expression. Also, it can perform the specified action (like deleting, printing, changing to some other pattern, enciphering, etc.) when the specified pattern is matched. It does this by converting regular expression into a non deterministic finite state `automata`- then a finite state automata—then reduces the number of states in it. lex is a program generator, which means we can write our own code, which functions the same as the `lex` output. Since lex applies a general set of rules to achieve this, what it generates will not make efficient use of memory and is slower too. Yet it is a powerful tool, that simplifies the programmer's job.

84. Two mistakes. First is the `?`. It is a meta character. So, when the shell encounters `?`, it will try for a match, with the files in the current directory, made up of just one character. Use `\?`, to suppress the special meaning of `?`. `$answer` means the value of the variable answer. Since you are reading the value of the variable answer, it should be read answer.

88. File descriptors of `stdin`, `stdout`, `stderr` are `0`, `1` and `2` respectively. `2>&1`, instructs the shell to merge the stream `stderr` with `stdout`. `1>&2`, merges `stdout` with `stderr`. This instruction is valid only for that line in the shell script. Also, if you use redirection in the command line itself, the `>&` takes precedence, if it contradicts.

89. There is a facility that allows shell scripts to take input from its own contents, e.g., `grep $1<<tillhere`, used in a shell script file `x`. Input to `grep` is that part of `x`, from the first character to the first occurrence of `tillhere` (in the beginning of the line). If `tillhere` is never encountered as the first word, the entire file `x`, will be taken as input.

90. `cc x.c; a.out`- means execute the command `cc x.c` and then `a.out`. If `x.c` fails to compile successfully, then if there is any executable file `a.out`, it will be executed. So, execution of `a.out`, has nothing to do with the outcome of `cc x.c`. In the case of `cc x.c && a.out`,`a.out` will be executed only if `x.c` compiles successfully (i.e. `re-turns 0` as the exit status).

100. * is a metacharacter that matches with any file in the current directory, other than those starting with a `.(dot)`. `?x` can't be a file name. If you try to create such a file, say with `vi ?x` command, ? will be interpreted as a metacharacter, and so expanded by shell, if matched.

102. `stdin` is a pointer to the standard input file (i.e. keyboard by `default`) which is available to any program in open mode. So, `getc(stdin)` is syntactically correct and means reading from a keyboard which is what `getchar()` does. In fact `getc()` is implemented as a macro (rather than as a function).

105. All the library `i/o` functions (like `getchar, scanf, gets` etc.) use the same intermediate buffer and share the same file pointer. So, they can be interleaved in any order to access consecutive characters in a file without causing any inconsistency. Unlike them, system calls (like `read` and `write`) directly manipulate the file. So, mixing system calls and library function will have undesired consequence.

106. The close statement closes the file, whose file descriptor is `1`, i.e., `stdout`. So, `printf` will fail. So, the program immediately terminates.

108. `return` statement when executed transfers control back to the calling environment. So, if a subroutine executes a `return` statement control comes back to the main routine. `exit` always terminates the program, which means within the main routine, `exit` and `return` can be used interchangeably.

116. Kernel should be involved anyway. `/etc/init`, which has to initiate all processes, forks and executes `/etc/getty`. It is `/etc/getty` that displays the `login prompt`. When one responds to the `login prompt`, the `/bin/login` replaces `getty`. Ultimately, the `login` shell replaces the `login` process.

117. `read` needs three arguments. The first argument should be a file descriptor `fd`. The standard input, `stdin` has the `fd` value `0`. The second argument should be a pointer to character, which is `buff` by our assumption. The third argument is the number of bytes to be read. So it is `read(0,buff,1)`.

129. Any error has an integer code associated with it. The external variables `errno` (integer), `sys_errlist` (array of strings), `sys_nerr`, are available to any C program. Any error sets `errno` to the associated integer code and `sys_errlist[errno]` gives the associated message. This way `perror ()` can be simulated by using `errno` and `sys_errlist`. `sys_nerr` gives the total number of error messages available in `sys_errlist`.

133. `(x +)`, means `x` followed by one or more number of blanks, `'*'` is a metacharacter which means the occurrence of `0` (i.e., not occurring even once) or any number of times of the proceeding pattern. So, `(x +)*` means the pattern `x +`, can occur `0` or any number of times. `(x +)*y!`-means the pattern `(x +)*` immediately followed by a `'y'` and `'!'`. `'$'` is a metacharacter, which means at the end of the line. So, choice (a) is wrong. So, `(x +)*y!$`, means `(x +)*y!` at the end of the line (i.e. last part of the line). Hence the answer is (d).

136. `tr`, unlike the other three, cannot access a named file for input. So, to make it access a file, say `x`, we have to use the redirection operator `<`. i.e. `tr` (action field) `< x`.

138. When a process executes a `fork()` statement, a duplicate process is created and both the processes execute all the statements following the `fork()` statement. So, the parent process when it executes the first fork statement creates a duplicate process. So, we have two processes both of which, will execute the statements `printf("2"); fork(); fork() printf("3");` So 2 will be printed twice. The next `fork()` call will produce a total of four processes (since it will be executed by the two existing processes each of which creates a duplicate process). So, the last `fork()` call, will produce eight processes, all executing `printf("3")`. Hence, 3 will be printed 8 times. Hence the correct answer is (b).

141. `expr` is a shell command that evaluates the arithmetic expression, given as argument to it. The multiplication symbol `*`, will be treated as a metacharacter by the shell. So, `2*3`, will not be interpreted as multiplying 2 and 5. The shell will expand it to include all the files in the current directory, starting with 2 and ending with 3. So, the correct answer is (c).

146. If a shell variable, say `x` is undefined, `echo $x` will display an empty line (this is the case, if `x` is set to a null string also). `$sh`, results in the current shell, forking a child shell and waiting for it to terminate. So, `x=hai`, will be executed only by the child shell and hence will not be recognized by the parent shell. As a result of the next command `$export x`, `x` will be passed to any shell forked by the child shell. The next command `$sh`, results in the child shell forking another shell (let us call it as the grand-child shell) and waiting for it to terminate. So, `echo $x` will be executed by the grand-child shell, which has inherited `x` and its value from the child shell because of the export command used. So, `echo $x` will display `hai`.

147. Execution of the command `exit`, results in the immediate termination of the current shell. In such a case, control will go to its parent shell. So, using `exit` twice, takes us back to the shell that displayed an empty line for `echo $x`. So, `echo $x` if run after two `exit` will display an empty line. So, the answer is (c).

148. When a process executes a `fork()` statement the duplicate process that is created survives as a separate and independent process. Because of the CPU time slice, either of the two may terminate first. If the parent terminates first, the forked process will be immediately adopted by the process dispatcher. If you run the `ps -al` command, this is reflected by a `O` in the status column.

149. Any call has to be made by some process. Any process will have an identity number. `getpid()`, returns this number. So, it can never fail. Hence the result.

150. Refer Question 138.

Execution of the first `fork()`, results in a total of two processes. The next `fork()`, makes it four. All the four processes, execute the statement `printf("yes");` So, `yes` will be printed four times.

151. Any process starts executing in user mode. If a system call is made (by executing a special machine instruction), the mode changes from user to kernel. Then the process executing in kernel (unrestricted) mode, services the call. After servicing, another context switch puts it back to the user mode. The hardware views a process by its mode (not by its PID).

152. Kernel is not a separate process that runs along with other processes. It is a part and parcel of other user processes. That is why we say system calls are serviced by a process executing in kernel mode. The kernel, as such, is not at all involved in servicing the system call. When a process making a system call has to wait for an event to happen, then first a context switch transforms the process from user to kernel mode. Then a process switch is initiated that switches control to the kernel process. After that event occurs, again a process switch followed by a context switch, puts the process back in the user mode and the execution is continued.

153. Before listing, it opens the file `xy` in the current directory. So, `cat xy` will display `xy` also. If `xy` already exists, its contents will be discarded and a new `xy` is created. However, if `xy` already exists and is `write` protected, an error message will be displayed on the screen.

155. `cc` command first invokes the C preprocessor `/lib/cpp`. The output will be redirected to `/lib/ccom` and `/lib/c2`. Then the assembler `/bin/as` is invoked. Finally the linking/ loader `ld` will be called. So, a total of five passes. Only `/lib/ccom` is actually 'C' language dependent.

161. Refer Qn. 155.

170. A `fork` call duplicates any process that executes it. But the `exec` family of calls, overlays the address space of the process that executed it. So, no chance of getting back to it.

171. The `wait()` function takes an argument that is a pointer to an integer. It fills the two lower order bytes of the integer that is pointed to, by its argument, with some information. A normal termination sets the lowest byte to zero and the other byte is filled with an integer that equals the `exit` status of the process, for which it was waiting. In case of abnormal termination, the second lowest byte is filled with zeroes and the lowest byte filled with an integer that reflects the cause of the abnormality (e.g. signal number). In case of a core dump or a zombie process, the seventh bit is set on.

172. Two things should happen before one sees a message on the screen. First, there should be a program that writes that message to a buffer. Secondly, the contents of the buffer should be sent to the screen. The buffer is necessary to balance the speed mismatch among different communicating devices. This program will put `WHATIZIT` in the buffer. Transfer from buffer to the terminal takes place if the buffer is full or `'\n'` is present in the message or `fflush(stdout)` is used explicitly. Since none of these cases applies to this case, it will not be transferred. So, the command `date` will be executed and displayed in the screen. Then the process terminates. But before termination, the buffer contents will be flushed out to the screen by default. So, `WHATIZIT` will be displayed, but after the output of the `date` command.

173. Refer Qn. 172.

The commands (like `date`), automatically feed `'\n'` on termination. So the given program prints `x` and output for `date` in the same line. So `a` is wrong. Hence the answer is (b).

174. `read()` does not check, whether the file is locked or not. So it successfully reads from a locked file. Same is the case with `write()`.

175. The function `main` can take up the three arguments. They are `argc` - an integer that gives the number of arguments, `argv`- an array of pointer to character, that gives the arguments, and `envp` - an array of pointer to character, that gives all the variables defined in the environment.

177. Each user executing the same executable file will be using a copy of it. This is a pure overhead, if the executable code is shareable (like `ed`, `vi`, etc.). By setting, what is called as

the sticky bit on, the executable code can be shared. In such a case, the file will be loaded into a special area called swap area. The sticky bit can be set by the command
`chmod u+t <filename>`

178. Each `fork` call creates a duplicate process. So if `n` is `1`, `fork` will be called only once. So two processes will be executing `printf("yes")`. So, `yes` will be printed twice. If `n=2`, four processes and hence 4 `"yes"` will be printed. If `n=3` then 8 `yes`, 4 then 16, 5 then 32, which means 24 `"yes"` is impossible.

179. Refer Qn. 172
Transfer from buffer to a terminal, usually happens on a line by line basis. This is called line buffering. That's why a `'\n'` forced a buffer flushing. But redirection to a file involves block buffering. Only a full buffer or program termination or `fflush()` will initiate the transfer from buffer to the named file. So, when the system command is executed, the current contents of the buffer (i.e., the first message) will not be transferred. On termination, the contents of the buffer, which has both the messages will be flushed out to the file.

180. `fork` returns 0 to the child process and a non-zero integer (i.e. the PID of the created child) to the parent. Hence the child process will immediately get terminated as it has nothing to execute. But the parent process will be active for at least 100 seconds (because of `sleep(100)`). Though the child terminates before the parent, the corresponding entry in the process table will not be removed till the termination of the parent process. For this reason it is called a zombie process.

183. Though the parent process has to wait for the termination of the child process (because of the `wait()` statement), the value updated by the child will not be recognized by the parent. So, it prints `7`.

184. The booting process loads `/unix` (i.e. kernel) into memory by some firmware and software operations. All other processes will be descendants of the process `/etc/init` which is created by the kernel exclusively for that purpose.

186. Refer to Qn. 183.
Even a pointer variable, which is used to change a value, by the child process, will not be recognized by the parent (& vice-versa). Anyway, the statement `printf("%d%d", &j, &i)` will produce identical output by both the processes, which implies that they use the same address space (of course at different times)

187. Even a global variable will not be shared.

189. `signal(SIGINT, mn)` is an instruction to execute the user defined function `mn` (instead of the `default` function), on receipt of the signal `SIGINT`. However, a second `SIGINT` will execute the `default` function.

190. Repeating the statement thrice, or any number of times, is equivalent to having only one statement, in essence. So the second SIGINT signal, will execute the default code.

191. Since its value has to be used by different processes, it is kept in the kernel.

192. The `ampersand`, runs the command as a background process. So, the three commands run as separate processes, each competing individually for CPU time slice. If they need more than one CPU time slice, the commands will overlap in their execution (with respect to time). So, unpredictable things can happen. For example, the first process creates the file `mn`.

Before termination if CPU switches over to the second process, then `wc` will act on file `mn`, whose data is not full. So, choice `a` is incorrect. Choices `b` and `c` are syntactically incorrect, as redirection can be done only to a file and not a command like `wc -l`. Hence the answer is `d`.

199. Refer Questions 189 and 190.

The two `fork` calls will result in a total of four processes, all of which execute `mn()` on the receipt of signal `SIGINT`. So, the signal, if received (by pressing the `<del>` key), will be sent to all the four processes. So, four `"x"` will be printed. One more `SIGINT` will do the `default` action of terminating the process.

200. The `read` and `write` ends of the pipe will be passed on to the child process. So, there will be four file descriptors in memory, albeit, a pair being duplicated. As a result of this duplication, one process may close one end, while the other may use the same as an open end.