Student ID: _____
Student Name: _____

Dhirubhai Ambani Institute of Information & Communication Technology
Second In-Semester Examination, Winter Semester 2022-2023

Course Title IT628 Systems Programming **Max Marks** 50
Date 28th Mar 2023 **Time** 1.5 Hours

Q1a(2)	Q1b(2)	Q1c(2)	Q1d(2)	Q1e(2)	Q1f(2)	Q1g(2)	Q1h(2)	Q1i(2)	Q1j(2)	Q1k(2)	Q2a(8)	Q2b(6)	Q3a(7)	Q3b(7)

Instructions:
1. All questions are compulsory.
2. Write the answers for all questions in the space provided in question paper only.
3. Write your answers in brief, writing long answers does not fetch higher marks.

Q1. Short Answer 1-3 lines [2 Marks each]

a. Upon executing fork(), code and data area of parent and child processes look the same however context look different. Explain briefly why?

Answer:
context consist of program control block (PCB) that will have fields of process id (pid) , parent process id (ppid) etc which will be different for parent and the child process. Therefore context look different but code and data area does not.

b. Once the following code is executed, specify how many entries will be created in system-wide file open table and what will be the values of offset and reference count for each entry. Assume that file1.txt exist with the file size of 100 bytes.

```
char buf[100];
int fd1=open("file1.txt", "O_RDWR ");
int fd2=open("file1.txt", "O_RDWR ");
write(fd1 , buf, 10);
read(fd2, buf, 20);
```

Answer:

file descriptor	offset	reference count
fd1	10	1
fd2	20	1

c. What will be the output from the code below? Explain briefly.

```
int x=10;
printf("%d\n",x);
fork();
int y=x + 5;
printf("%d %d\n",x,y);
```

Answer:
10
10 15
10 15

x declared and printed before fork will print the first line with value of x (10) only once in the main process. Variable x is duplicated after fork() in both parent and child processes. Variable y is declared after fork() so both parent and child will have variable y and will be initialized with value 15 (x + 5). The printf() after fork will be executed by both processes and hence will print "10 15" in line 2 and 3 as the values of x and y.

d. When performing "ls -i" command in current directory following content is displayed.

1000 file1.txt
1000 file2.txt
1001 file3.txt -> file1.txt

We execute the three commands "rm file1.txt", "cat file2.txt", "cat file3.txt" in the same order. What will be the output from these commands? Explain in brief.

Answer:

file1.txt and file2.txt are hardlinks and file3.txt is a softlink to file1.txt. After deleting file1.txt, softlink file3.txt will become invalid therefore "cat file3.txt" will fail where "cat file2.txt" will succussed because deleting one hardlink does not affect the other hardlink.

e. What will be the output from the code below? Explain briefly.

```
int x=10;
printf("%d\n",x);
if (fork() == 0)
   int y = 5;
int z=x+y;
printf("%d %d %d\n",x,y,z);
```

Answer:

Compile time error "undefined variable" will be generated because variable y is defined only for child process but z=x+y is executed for both parent and child process and parent process does not have y variable defined.

f. What is the difference between /bin and /usr/bin directories? Briefly explain with an example why both are required.

Answer:

/bin folder contain system level binaries accessed by all user where /usr/bin contains the binaries specific to a particular reason. For example, gcc is installed for the first time, say version 8, the binaries will be in /bin whereas when new a version, say version 9, is installed by current user the binaries will be installed in /usr/bin. Therefore when other user accessed gcc they will be able to access it using /bin folder gcc version 8 and current user will be able to access version 9 from /usr/bin

g. Considering the code below what are sequence of states a child process goes through while reading a complete file.txt? Assume that file.txt has 15 bytes of data and possible process lifecycle states are idle, runnable, running, sleeping, suspended and zombified.

```
int fd = open("file.txt", O_RDONLY);
if (fork() == 0) {
   int bytes_read=100;
   while (bytes_read>=10)
      bytes_read = read(fd, buf, 10);
}
```

Answer:

idle, (runnable running, sleeping) → reading first 10 bytes, (runnable, running, sleeping) → reading last 5 bytes, runnable, running

h. Explain briefly the output from the code assuming that file.txt has "-r--r--r--" permission.

```
struct stat st;
stat("file.txt", &st);
printf("%d", st.st_mode & S_IRUSR);
```

Answer:

The output will be 1 i.e. true because stat function returns status of the file and by binary AND operation between st.st_mode and S_IRUSR we can check if the current user has read permission to the file which it has.

i. Considering a code below, draw a binary tree to show how many processes will be created? Assume that process id of main process starts with 10 and each subsequently created processes have process ids incremented by 1 i.e. 11, 12, 13 etc.

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

Answer:

(fork() i=0) - 10 (fork() i=1) - 10 (fork() i=2) – 10
13
12 (fork() i=2) - 12
14
11 (fork() i=1) - 11 (fork() i=2) - 11
16
15 (fork() i=2) - 15
17

Total 8 processes

j. Once the following code is executed, specify how many entries will be created in system-wide file open table and what will be the values of offset and reference count for each entry. Assume that file1.txt exist with the file size of 100 bytes.

```
char buf[100];
int fd1=open("file1.txt", "O_RDWR ");
int fd2=dup(fd1);
read(fd1 , buf, 20);
write(fd2, buf, 10);
```

Answer:

file descriptor	offset	reference count
fd1 and fd2	30	2

k. Considering a code below, draw a binary tree to show how many processes will be created? Assume that process id of main process starts with 10 and each subsequently created processes have process ids incremented by 1 i.e. 11, 12, 13 etc.

```
for (int i=0; i<2; i++) {
  fork();
  execl("/bin/ls", "/bin/ls", "-l");
}
```

Answer:

Only two (parent and child) from the first fork() execution because both parent and child will execute excel() which will replace the code from the current executable to code of ls.

Q2

a. As studied in class, the following code redirects standard output (stdout) to a file output.txt. Modify/Rewrite the code to copy the content of input.txt file to output.txt file using input and output redirection. [8 Marks]

```
char msg[] = "Dummy Message\n";
int fd = open("output.txt", O_WRONLY | O_CREAT, 777);
dup2( fd, 1);
write(1, msg, strlen(msg));
close(fd);
```

Answer:

```
int main()
{
        char buf[150];
        int bytes_read=0;
        //char msg[] = "Write something to stdout but it will be redirected to a file\n";
        int fd = open("input.txt", O_RDONLY);
        dup2( fd, 0);  /* Redirect stdin to a file*/
        int fd1 = open("copy_input.txt", O_WRONLY | O_CREAT, 777);
        dup2( fd1, 1);  /* Redirect stdout to a file*/
        while ((bytes_read = read(0, buf, 100)) > 0)
                write(1, buf, bytes_read);
        //write(1, msg, strlen(msg));
        close(fd);
}
```

b. Considering File Control Block (FCB) or inode supports 10 direct pointers, 10 single indirect pointers, 5 double indirect pointers and 2 triple indirect pointers. Calculate the maximum size of the file supported by this FCB when each of the data block address is 32 bits. Assume that each data block is of size 4KB (4096 bytes). [6 Marks]

Answer:

10 direct pointers = 10*4KB = 40 KB
10 single indirect pointers = 10*4096/32*4KB = 30120 KB
5 double indirect pointers = 5*4096/32*4096/32*4KB = 327680 KB
2 triple indirect pointers = 2*4096/32*4096/32*4096/32*4KB = 16777216 KB
Total = 17,135,056 KB or 16,733.453125 MB or 16.3412628173828125 GB

Q3

a. Considering a code below, draw a binary tree to show how many processes will be created? Assume that process id of main process starts with 10 and each subsequently created processes have process ids incremented by 1 i.e. 11, 12, 13 etc. [7 Marks]

```
fork();                 // line1
if (fork() > 0)         // line2
   fork();              // line 3
fork();                 // line 4
```

Answer:

	1 Mark	**3 Marks**	**3 Marks**	
(fork() line 1) -	10 (fork() line 2)	12 (fork() line 3) -	12 (fork() line 4) –	12

			17
		15 (fork() line 4) -	15
			18
	13	(fork() line 4) -	13
			19
11 (fork() line 2) -	11 (fork() line 3) -	11 (fork() line 4) -	11
			20
		16 (fork() line 4) -	16
			21
	14	(fork() line 4) -	14
			22

Total 12 processes

b. We have a 1000 elements array a[1000] representing bitmap of disk blocks availability. Each element will have 1 if a corresponding disk block is available and 0 if disk block is unavailable. We need to count how many total disk blocks are available. To speed up the counting, we will create 5 child processes each will count 200 values i.e. first child process will count 1's for indices 0 to 199, second child will count 1's in indices 200 to 399 etc. Each child process will return the number of 1's counted as exit code which parent process will read to calculate the total count. Please fill in the code to achieve this task. [7 Marks]

Answer:

```
int a[1000];      // array containing bit map
int cnt=0;        // will be used to count the number of 1's
int pid, childpid, status;
main() {
  for (int i=0, i<5; i++) {
    pid = fork();
    // child part
    if (pid == 0) {
      for (int j=i*200; j< i*200+200; j++)
        if (a[[j] == 1)
          cnt++;
      exit(cnt);
    }
```

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

```
    // parent part
```

```
      else {
         childpid = wait(&status);
         cnt = cnt + WEXITSTATUS(status);
```

```
      }
   }
}
```