Unix System Programming Lab Internals questions - Jan- May 2018

Sl. No.	Question	Course Outcome
1.	Write a C program to display the file content in reverse order using lseek system call.	CO1
2.	Write a C program a. to read first 20 characters from a file b. seek to 10th byte from the beginning and display 20 characters from there c. seek 10 bytes ahead from the current file offset and display 20 characters d. display the file size	CO1
3.	Write a C program to implement the ls –li command which list the files in a specified directory. Your program should Print 5 attributes of files.	CO1
4.	Write a C program to demonstrate the creation of soft links and the various properties of hard links	CO1
5.	Write a C program to Copy access and modification time of a file to another file using utime function.	CO1
6.	Write a C program to illustrate the effect of setjmp and longjmp functions on register and volatile variables.	CO2
7.	C program to simulate copy command by accepting the filenames from the command line. Report all errors.	CO1
8.	Write a C program to avoid the zombie status of a process.	CO2
9.	Write a C program to demonstrate race conditions among parent and child processes.	CO2
10	Write a C program such that it initializes itself as a daemon Process.	CO2
11.	Write a C program using sigaction system call which calls a signal handler on SIGINT signal and then reset the default action of the SIGINT signal	CO3
12	Write a C program (use signal system call) i. which calls a signal handler on SIGINT signal and then reset the default action of the SIGINT signal ii. Which ignores SIGINT signal and then reset the default action of SIGINT signal	CO3

1. Write a C program to display the file content in reverse order using lseek system call.

```
#include <unistd.h>

#include <fcntl.h>

int main(int argc,char* argv[])

{

   char ch;

   int fd = open(argv[1], O_RDONLY);

   int k = -1;

   int beg = lseek(fd, 0, SEEK_CUR);

   int offset = lseek(fd, k, SEEK_END);

   while (offset>=beg)

   {

      read(fd,&ch,1);

      write(1,&ch,1);

      k--;

      offset = lseek(fd, k, SEEK_END);

   }

   return 0;

}
```

./a.out file.txt

2.Write a C program

a. to read first 20 characters from a file
b. seek to 10th byte from the beginning and display 20 characters from there
c. seek 10 bytes ahead from the current file offset and display 20 characters
d. display the file size

```
#include <unistd.h>

#include <fcntl.h>
```

```
#include <string.h>
#include <stdio.h>
int main(int argc,char* argv[])
{
   char size[10];
   int k;
   int fd = open(argv[1], O_RDONLY);
   char buffer[20];
   write(1,"Read first 20 bytes\n",20);
   read(fd,buffer,20);
   write(1,buffer,20);
   write(1,"Seek to 10th byte from beginning and read 20 bytes\n",51);
   lseek(fd,10,SEEK_SET);
   read(fd,buffer,20);
   write(1,buffer,20);
   write(1, "Seek to 10th byte from current offset and read 20 bytes\n", 56);
   lseek(fd,10,SEEK_CUR);
   read(fd, buffer, 20);
   write(1, buffer, 20);
   write(1, "Size of the file\n", 17);
   k = lseek(fd,0,SEEK_END);
   sprintf(size,"%d",k);
   write(1,size,strlen(size));
}
./a.out file.txt
```

3. Write a C program to implement the ls –li command which list the files in a specified directory. Your program should Print 5 attributes of files.

```
#include<stdio.h>
#include <fcntl.h>
#include <dirent.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include<time.h>
int main(int argc, char *argv[])
{
   DIR *dp; struct dirent *dirp;
   struct stat sb;
   dp = opendir(".");
   while ((dirp = readdir(dp)) != NULL)
   {
      if(lstat(dirp->d_name, &sb) == -1)
      {
         perror("lstat");
         exit(0);
      }
      printf("%s\n",dirp->d_name);
      printf("I-node number:%ld\n", (long) sb.st_ino);
      printf("Mode:%lo (octal)\n",(unsigned long) sb.st_mode);
      printf("Link count:%ld\n", (long) sb.st_nlink);
```

```
      printf("Ownership:UID=%ld   GID=%ld\n",(long) sb.st_uid, (long) sb.st_gid);
      printf("Preferred I/O block size: %ld bytes\n",(long) sb.st_blksize);
      printf("File size:%lld bytes\n",(long long) sb.st_size);
      printf("Blocks allocated:%lld\n",(long long) sb.st_blocks);
      printf("Last status change:%s", ctime(&sb.st_ctime));
      printf("Last file access:%s", ctime(&sb.st_atime));
      printf("Last file modification:%s", ctime(&sb.st_mtime));
      printf("\n\n");
   }
   closedir(dp);
   return 0;
}
./a.out
```

4. Write a C program to demonstrate the creation of soft links and the various properties of hard links

```
/* Write a C program to demonstrate the creation of soft links and the various properties of hard links */

#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
int main(int argc, char *argv[])
{
  if (argc == 3)
```

```
 {
   printf("Hard linking %s and %s", argv[1], argv[2]);
   if (link(argv[1], argv[2]) == 0)
     printf("\nHard link created");
   else
     printf("\nLink not created");
 }
 else if (argc == 4)
 {
   printf("Soft linking %s and %s", argv[1], argv[2]);
   if (symlink(argv[1], argv[2]) == 0)
     printf("\nSoft link created");
   else
     printf("\nLink not created");
 }
}
```

for hardlink - ./a.out file.txt hard.txt

for soft - ./a.out file.txt ex.txt soft.txt

5.Write a C program to Copy access and modification time of a file to another file using utime function.

```
/* Write a C program to Copy access and modification time of a file to another file using utime function.
*/

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>
```

```
#include <unistd.h>
#include <utime.h>
#include <time.h>
#include <fcntl.h>

int main(int argc, char *argv[]) //copying ctime and mtime of argv[2] to argv[1]
{
  int fd;
  struct stat statbuf_1;
  struct stat statbuf_2;
  struct utimbuf times;

  if (stat(argv[1], &statbuf_1) < 0)
    printf("Error!\n");
  if (stat(argv[2], &statbuf_2) < 0)
    printf("Error!\n");
  printf("Before Copying ...\n");
  printf("Access Time %s\nModification Time%s\n", ctime(&statbuf_1.st_atime),
  ctime(&statbuf_1.st_mtime));

  times.modtime = statbuf_2.st_mtime;
  times.actime = statbuf_2.st_mtime;

  if (utime(argv[1], &times) < 0)
    printf("Error copying time \n");
```

```
 if (stat(argv[1], &statbuf_1) < 0)

   printf("Error!\n");


 printf("After Copying ...\n");


  printf("Access Time %s\nModification Time%s\n", ctime(&statbuf_1.st_atime),
ctime(&statbuf_1.st_mtime));

}

./a.out file1.txt file2.txt
```

6.Write a C program to illustrate effect of setjmp and longjmp functions on register and volatile variables.

```
/* Write a C program to illustrate effect of setjmp and longjmp functions on register and volatile
variables. */

#include <setjmp.h>

#include <stdio.h>

#include <stdlib.h>

static void f1(int, int, int, int);

static void f2(void);

static jmp_buf jmpbuffer;

static int globval;

int main(void)

{

 int autoval;

 register int regival;

 volatile int volaval;

 static int statval;
```

```
  globval = 1;
  autoval = 2;
  regival = 3;
  volaval = 4;
  statval = 5;
  if (setjmp(jmpbuffer) != 0)
  {
    printf("after longjmp:\n");
    printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n", globval, autoval, regival,
volaval, statval);
    exit(0);
  }
  /*
    Change variables after setjmp, but before longjmp.
  */
  globval = 95;
  autoval = 96;
  regival = 97;
  volaval = 98;
  statval = 99;
  f1(autoval, regival, volaval, statval); /* never returns */
  exit(0);
}
static void f1(int i, int j, int k, int l)
{
```

```
 printf("in f1():\n");
 printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n", globval, i, j, k, l);
 globval = 10000;
 j = 10000;
 f2();
}
static void f2(void)
{
 longjmp(jmpbuffer, 1);
}
./a.out
```

7. C program to simulate copy command by accepting the filenames from command line. Report all errors.

```
/* Write a  C program to simulate copy command by accepting the filenames from command line. Report all errors */
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
 char buf[100];
 int fd1, fd2;
 off_t size, ret, set;
 ssize_t readdata, writedata;
```

```
if (argc < 3)
        {
  printf("TOO FEW ARGUMENTS");
        exit(1);
        }
fd1 = open(argv[1], O_RDONLY); //Open file 1
if (fd1 == -1)
  printf("ERROR IN OPENING FILE: FILE DOES NOT EXIST \n");
else
  printf("FILE 1 OPENED SUCCESSFULLY \n");

 fd2 = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0666); //open file 2 in read-write mode, truncate its length to 0, create the file if it does not exist, 0666 is the access permission for the created file. order is important.
if (fd2 == -1)
  printf("ERROR IN OPENING FILE");
else
  printf("FILE 2 OPENED SUCCESSFULLY \n");

size = lseek(fd1, 0L, SEEK_END); //obtain the size of file 1 using lseek
if (size == -1)
  printf("ERROR: COULD NOT OBTAIN FILE SIZE \n");
else
  printf("FILE SIZE OF FILE 1 OBTAINED \n");

ret = lseek(fd1, 0L, SEEK_SET); //change the current pointer to the beginning of the file
```

```
 if (ret == -1)
   printf("RETRACE FAILED \n");

 readdata = read(fd1, buf, size); //read data equal to the size of the first file
 if (readdata == -1)
   printf("ERROR IN READING FILE CONTENTS \n");

 writedata = write(fd2, buf, size); //write the data to file 2 from buffer after read
 if (writedata != size)
   printf("ERROR IN COPYING FILE");
 else
   printf("FILE COPIED SUCCESSFULLY");
 return 0;
}
./a.out file1.txt file2.txt
```

8.Write a C program to avoid zombie status of a process.

```
/*  Write a C program to avoid zombie status of a process. */
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include<unistd.h>
int main(void)
{
  pid_t pid;
  if ((pid = fork()) < 0)
```

```
  {
    printf("fork error"); exit(0);
  }
  else if (pid == 0)
  {
    /* first child */
    if ((pid = fork()) < 0)
    {
      printf("fork error");
      exit(0);
    }
    else if (pid > 0)
      exit(0);
    sleep(2);
    printf("second child, parent pid = %ld\n", (long)getppid());
    exit(0);
  }
  if (waitpid(pid, NULL, 0) != pid)
    printf("waitpid error");
  exit(0);
}
./a.out
```

9.Write a C program to demonstrate race condition among parent and child processes.

```
/* Write a C program to demonstrate race condition among parent and child processes */
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include<unistd.h>
static void charatatime(char *);
int main(void)
{
  pid_t pid;
  if ((pid = fork()) < 0)
  {
    printf("fork error");
  }
  else if (pid == 0)
  {
    charatatime("output from childccccccccccccccccccccccccccccc\n");
  }
  else
  {
    charatatime("output from parentppppppppppppppppppppppp\n");
  }
  exit(0);
}
static void charatatime(char *str)
```

```
{
  char *ptr;
  int c;
  setbuf(stdout, NULL); /* set unbuffered */
  for (ptr = str; (c = *ptr++) != 0;)
    putc(c, stdout);
}
./a.out
```