Week 11 – Assignments

1. Write a C program to demonstrate file locking (A/P).

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

int main() {

struct flock fl;
int fd;

fl.l_type   = F_WRLCK;  /* read/write lock */
fl.l_whence = SEEK_SET; /* beginning of file */
fl.l_start  = 0;        /* offset from l_whence */
fl.l_len    = 0;        /* length, 0 = to EOF */
fl.l_pid    = getpid(); /* PID */

fd = open("locked_file", O_RDWR | O_EXCL); /* not 100% sure if O_EXCL needed */

fcntl(fd, F_SETLKW, &fl); /* set lock */

usleep(10000000);

printf("\n release lock \n");

fl.l_type   = F_UNLCK;
fcntl(fd, F_SETLK, &fl); /* unset lock */

}
```

2. Write a C program to demonstrate the function of a pipe (A/P).

```
// C program to demonstrate use of fork() and pipe()

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

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

```
#include <unistd.h>

int main()
{
    // We use two pipes
    // First pipe to send input string from parent
    // Second pipe to send concatenated string from child

    int fd1[2]; // Used to store two ends of first pipe
    int fd2[2]; // Used to store two ends of second pipe

    char fixed_str[] = "forgeeks.org";
    char input_str[100];
    pid_t p;

    if (pipe(fd1) == -1) {
        fprintf(stderr, "Pipe Failed");
        return 1;
    }
    if (pipe(fd2) == -1) {
        fprintf(stderr, "Pipe Failed");
        return 1;
    }

    scanf("%s", input_str);
    p = fork();
```

```
if (p < 0) {
        fprintf(stderr, "fork Failed");
        return 1;
}

// Parent process
else if (p > 0) {
        char concat_str[100];

        close(fd1[0]); // Close reading end of first pipe

        // Write input string and close writing end of first
        // pipe.
        write(fd1[1], input_str, strlen(input_str) + 1);
        close(fd1[1]);

        // Wait for child to send a string
        wait(NULL);

        close(fd2[1]); // Close writing end of second pipe

        // Read string from child, print it and close
        // reading end.
        read(fd2[0], concat_str, 100);
        printf("Concatenated string %s\n", concat_str);
        close(fd2[0]);
}

// child process
else {
        close(fd1[1]); // Close writing end of first pipe

        // Read a string using first pipe
        char concat_str[100];
        read(fd1[0], concat_str, 100);

        // Concatenate a fixed string with it
        int k = strlen(concat_str);
        int i;
        for (i = 0; i < strlen(fixed_str); i++)
                concat_str[k++] = fixed_str[i];

        concat_str[k] = '\0'; // string ends with '\0'

        // Close both reading ends
        close(fd1[0]);
        close(fd2[0]);

        // Write concatenated string and close writing end
        write(fd2[1], concat_str, strlen(concat_str) + 1);
        close(fd2[1]);
        exit(0);
    }
}
```

3. Write a C program for demonstrating pipe function using dup system call (A/P).

```
// C program to illustrate
// pipe system call in C
#include <stdio.h>
#include <unistd.h>
#define MSGSIZE 16
char* msg1 = "hello, world #1";
char* msg2 = "hello, world #2";
char* msg3 = "hello, world #3";

int main()
{
    char inbuf[MSGSIZE];
    int p[2], i;

    if (pipe(p) < 0)
        exit(1);

    /* continued */
    /* write pipe */
```

```
        write(p[1], msg1, MSGSIZE);

        write(p[1], msg2, MSGSIZE);

        write(p[1], msg3, MSGSIZE);


        for (i = 0; i < 3; i++) {

                /* read pipe */

                read(p[0], inbuf, MSGSIZE);

                printf("% s\n", inbuf);

        }

        return 0;

}
```

4. Write a C program to demonstrates how a sender might setup a connection to FIFO and send a message (A/P)

```
/* Filename: fifoserver.c */
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FIFO_FILE "MYFIFO"
int main() {
  int fd;
  char readbuf[80];
  char end[10];
  int to_end;
  int read_bytes;

  /* Create the FIFO if it does not exist */
  mknod(FIFO_FILE, S_IFIFO|0640, 0);
  strcpy(end, "end");
  while(1) {
    fd = open(FIFO_FILE, O_RDONLY);
    read_bytes = read(fd, readbuf, sizeof(readbuf));
    readbuf[read_bytes] = '\0';
    printf("Received string: \"%s\" and length is %d\n", readbuf, (int)strlen(readbuf));
    to_end = strcmp(readbuf, end);
    if (to_end == 0) {
      close(fd);
      break;
    }
  }
  return 0;
}
```

5. Explain in detail and write C program to show function of sending and receiving messages in message queues (A/P)

```
// C Program for Message Queue (Writer Process)

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#define MAX 10


// structure for message queue

struct mesg_buffer {

        long mesg_type;

        char mesg_text[100];

} message;


int main()

{

        key_t key;
```

```
    int msgid;

    // ftok to generate unique key
    key = ftok("progfile", 65);

    // msgget creates a message queue
    // and returns identifier
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;

    printf("Write Data : ");
    fgets(message.mesg_text,MAX,stdin);

    // msgsnd to send message
    msgsnd(msgid, &message, sizeof(message), 0);

    // display the message
    printf("Data send is : %s \n", message.mesg_text);

    return 0;
}
```