Q1

// Run Consumer code first then the Producer code

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <limits.h>

#define FIFO\_NAME "/tmp/my\_fifo"

#define BUFFER\_SIZE PIPE\_BUF

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

{

int pipe\_fd;

int res;

int open\_mode = O\_RDONLY;

int buffer;

int bytes\_read = 0;

printf("Process %d opening FIFO O\_RDONLY \n", getpid());

pipe\_fd = open(FIFO\_NAME, open\_mode);

printf("Process %d result %d \n", getpid(), pipe\_fd);

if (pipe\_fd != -1)

{

res = read(pipe\_fd, &buffer, sizeof(buffer));

while(res > 0)

{

printf(" Hello %d \n",buffer);

res = read(pipe\_fd, &buffer, sizeof(buffer));

bytes\_read += res;

}

(void)close(pipe\_fd);

}

else

exit(EXIT\_FAILURE);

printf("Process %d finished %d bytes read \n", getpid(), bytes\_read);

exit(EXIT\_FAILURE);

return 0;

}

// Run Consumer code first then the Producer code

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <limits.h>

#include <sys/types.h>

#include <sys/stat.h>

#define FIFO\_NAME "/tmp/my\_fifo"

#define BUFFER\_SIZE PIPE\_BUF

#define TEN\_MEG (1024 \* 1024 \* 10)

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

{

int pipe\_fd;

int res;

int open\_mode = O\_WRONLY;

int bytes\_sent = 0;

int buffer;

if (access(FIFO\_NAME, F\_OK) == -1)

{

res = mkfifo(FIFO\_NAME, 0777);

if (res != 0)

{

fprintf(stderr, "Could not create fifo %s \n", FIFO\_NAME);

exit(EXIT\_FAILURE);

}

}

printf("Process %d opening FIFO O\_WRONLY \n", getpid());

pipe\_fd = open(FIFO\_NAME, open\_mode);

printf("Process %d result %d \n", getpid(), pipe\_fd);

if (pipe\_fd != -1)

{

int count=0;

while (count < 4)

{

scanf("%d",&buffer);

res = write(pipe\_fd, &buffer, sizeof(buffer));

if (res == -1)

{

fprintf(stderr, "Write error on pipe \n");

exit(EXIT\_FAILURE);

}

count++;

}

(void)close(pipe\_fd);

}

else

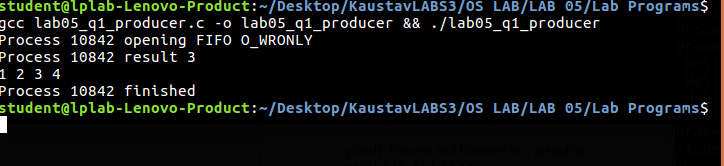
exit(EXIT\_FAILURE);

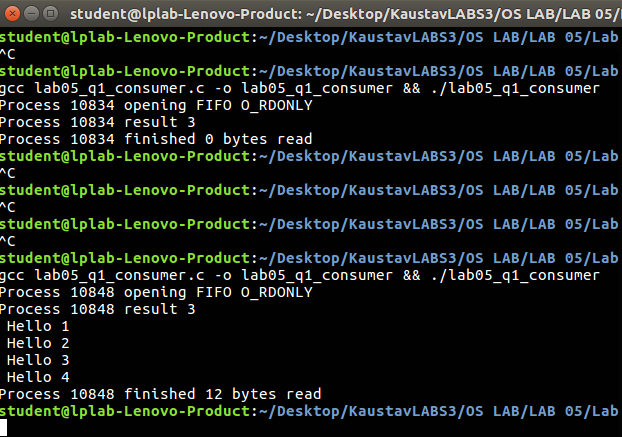
printf("Process %d finished \n", getpid());

exit(EXIT\_SUCCESS);

return 0;

}





Q2

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

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

{

int res = mkfifo("/home/student/Desktop/myfifo", 0777);

if (res == 0)

printf("FIFO CREATED \n");

else

printf("FIFO CREATION FAILED \n");

exit(EXIT\_SUCCESS);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <assert.h>

#include <unistd.h>

#include <sys/wait.h>

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

{

int pfd[2];

char buffer[1024];

pid\_t cpid;

assert(argc == 2);

pipe(pfd);

if (pipe(pfd) == -1)

{

perror("pipe");

exit(EXIT\_FAILURE);

}

cpid = fork();

if (cpid == -1)

{

perror("fork");

exit(EXIT\_FAILURE);

}

else if (cpid == 0)

{

close(pfd[1]);

while (read(pfd[0], &buffer, 1) > 0)

write(STDOUT\_FILENO, &buffer, 1);

write(STDOUT\_FILENO, "\n", 1);

close(pfd[0]);

exit(EXIT\_SUCCESS);

}

else

{

close(pfd[0]);

write(pfd[1], argv[1], strlen(argv[1]));

close(pfd[1]);

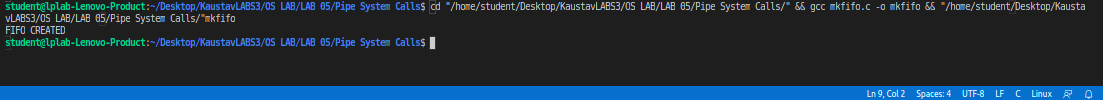
wait(NULL);

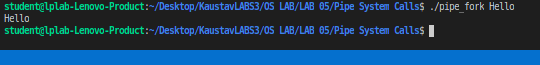
exit(EXIT\_SUCCESS);

}

return 0;

}





Q3

// C program to implement one side of FIFO

// This side writes first, then reads

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int fd;

// FIFO file path

char \*myfifo = "/tmp/myfifo";

// Creating the named file(FIFO)

// mkfifo(<pathname>, <permission>)

mkfifo(myfifo, 0666);

char arr1[80], arr2[80];

while (1)

{

// Open FIFO for write onlyi

fd = open(myfifo, O\_WRONLY);

// Take an input arr2ing from user.

// 80 is maximum length

fgets(arr2, 80, stdin);

// Write the input arr2ing on FIFO

// and close it

write(fd, arr2, strlen(arr2) + 1);

close(fd);

// Open FIFO for Read only

fd = open(myfifo, O\_RDONLY);

// Read from FIFO

read(fd, arr1, sizeof(arr1));

// Print the read message

printf("User2: %s\n", arr1);

close(fd);

}

return 0;

}

// C program to implement one side of FIFO

// This side reads first, then reads

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

int fd1;

// FIFO file path

char \*myfifo = "/tmp/myfifo";

// Creating the named file(FIFO)

// mkfifo(<pathname>,<permission>)

mkfifo(myfifo, 0666);

char str1[80], str2[80];

while (1)

{

// First open in read only and read

fd1 = open(myfifo, O\_RDONLY);

read(fd1, str1, 80);

// Print the read string and close

printf("User1: %s\n", str1);

close(fd1);

// Now open in write mode and write

// string taken from user.

fd1 = open(myfifo, O\_WRONLY);

fgets(str2, 80, stdin);

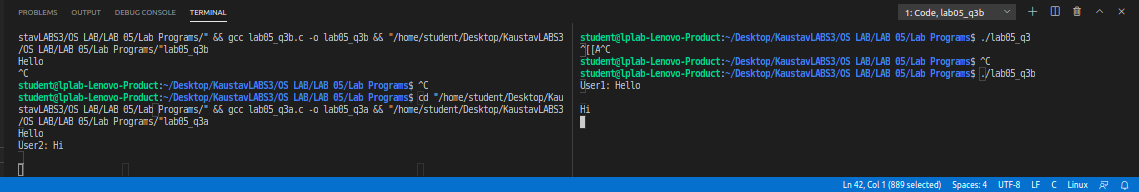
write(fd1, str2, strlen(str2) + 1);

close(fd1);

}

return 0;

}



Q4

#include <stdio.h>

#include <stdlib.h>

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

{

FILE \*fptr;

fptr = fopen("example.bin", "wb+");

int num1 = 0;

int num2 = 0;

for (int i = 1; i <= 4; i++)

{

scanf(" %d", &num1);

fwrite(&num1, sizeof(int), 1, fptr);

}

printf("Write Finsihed ! \n");

fclose(fptr);

fptr = fopen("example.bin", "rb");

for (int i = 1; i <= 4; i++)

{

fread(&num2, sizeof(int), 1, fptr);

printf(" %d \n", num2);

}

printf("Read Finished !!\n");

return 0;

}

