**Exercise1.1:**

a.Explain what the flags O _WRONLY , O_APPEND and O_CREAT do.

_RDONLY
Open for reading only.

O_WRONLY
Open for writing only.

O_APPEND
If set, the file offset shall be set to the end of the file prior to each write.

O_CREAT
If the file exists, this flag has no effect. Otherwise, the file shall be created.

b.Explain what the modes S_IRUSR, S_IWUSR do.

S_IRUSR-
Read permission bit for the owner of the file. On many systems this bit is 0400. S_IREAD is an obsolete synonym provided for BSD compatibility.
S_IWUSR-
Write permission bit for the owner of the file. Usually 0200. S_IWRITE is an obsolete synonym provided for BSD compatibility.

(http://man7.org/linux/man-pages/man2/open.2.html)

Exercise1.2:

DONE..

**Exercise2.1:**

Look at the code in example2.2.c and answer the following questions.
a. What does write(STDOUT_FILENO, &buff, count); do?

STDOUT_FILENO is defined in the system header file, unistd.h. It is the GNU/Linux file descriptor for standard out (usually the screen).buffer content will be written in to that file descriptor.

b. Can you use a pipe for bidirectional communication? Why (not)?

yes you can but it don't have to be.In particular, they aren't on Linux. Different processes are trying to read from and write to the same pipe.therefore there should be proper sysnchronization mechanism to manage read write access for processes if the pipe is bidirectional.Otherwise you have to use separate pipes for child and parent processes.

c. Why cannot unnamed pipes be used to communicate between unrelated processes?

Unnamed pips can only use to communicate between related processes. Because the processes are related, the association of file descriptors to the pipe can be implicit and does not require an object with a name that is external to the processes. An unnamed pipe exists only as long as the processess that use it maintain open file descriptors to the pipe. When the processes exit and the OS closes all of the file descriptors associated with the processes, the unnamed pipe is closed.therefore you cant use unnamed pipes for communication between unrelated processes that might not exist at the same time.

d. Now write a program where the parent reads a string from the user and send it to the childand the child capitalizes each letter and sends back the string to parent and parent displays it. You'll need two pipes to communicate both ways.

DONE in ex2.c

**Exercise3.1:**

because it replace the execution image of a forked child, all the communication means are lost since exec() replaces all the original code.Therefore using this pipes (unnamed pipes) cant communicate with parent without existance of childprocess.

**Exercise3.2:**

a. What does 1 in the line dup2(out,1); in the above program stands for?

The dup2() system call performs the same task as dup(), but instead of using the lowest-numbered unused file descriptor, it uses the file descriptor number specified in newfd.in here out is the newfd.it replace the stdout with the out file discriptor.whatever write to the stdout wll be in the "out".

b. The following questions are based on the example3.2.c

i.Compare and contrast the usage of dup() and dup2() . Do you think both functions are necessary? If yes, identify use cases for each function. If not, explain why?

dup => int dup(int oldfd); uses the lowest-numbered unused descriptor for the new descriptor.

dup2=> int dup2(int oldfd, int newfd); makes newfd be the copy of oldfd.

As explain above dup2 use newfd but dup use only the lowest numbered unused desciptor.therefore if you want to copy of file disciptor which is not the lowest-numbered then you should use the dup2 and you can use dup2 allways instead of dup.

ii.There’s one glaring error in this code (if you find more than one, let me know!). Can you identify what that is (hint: look at the output)?

program is not exit.parent exist before the child. hence child is looking more inputs.

iii. Modify the code to rectify the error you have identified above.

DONE in ex32b.c

**Exercise 4.1:**

a).

DONE in ex41reader.c and ex41writer.c

b).

DONE in prog1_42.c and prog2_42.c