Interprocess Communication

E/15/202

Exercise 1.1

a) O_WRONLY means opening file is “Write only”
O_APPEND means that the file is opened in append mode.
O_CREAT means, if pathname does not exist, create it as a regular file.

b) S_IRUSR says that the user has read permission
S_IWUSR says that the user has write permission

Exercise 1.2

a) mycat.c
b) mycopy.c

Exercise 2.1

a) It writes the content which is in the buffer to the standard output

b) We cannot use a single pipe for bidirectional communication. Pipes are unidirectional. We have to use 2 pipes for bidirectional communication.
If we use a single pipe for bidirectional communication then we have to keep opening the both ends of the pipe. So when a person writes at the write end then it will available for everyone at the read end. Since the write end of pipe is not closed the read end will not be able to identify the end-of-file. Because of that a single pipe is unidirectional.

c) Unnamed pipes are created with file 2 file descriptors as read end and write end. There is no way to identify those descriptors without using fork() command to inherit those descriptors from parent to child. File descriptors are used according to the inheritance. So they can only communicate with related processes.

d) ex21d.c

Exercise 3.1

This cannot be done. Because when we use exec(), it will replace all the processes including the parent processes(simply it replaces all the original code)by it’s shell command. So that all the communication means will be lost.

Exercise 3.2

a) It is the file descriptor number used for standard output

b) i)

- dup(int oldfd) system call creates a copy of the file descriptor.
- It uses the lowest-numbered unused descriptor for the new descriptor.
- If the copy is successfully created, then the original and copy file descriptors may be used interchangeably.
- They refer to the same open file description and thus share file offset and file status flags; for example, if the file offset is modified by using lseek(2) on one of the descriptors, the offset is also changed for the other.
- The two descriptors do not share file descriptor flags (the close-on-exec flag). The close-on-exec flag (FD_CLOEXEC;) for the duplicate descriptor is off.

- The dup2(int oldfd, int newfd) system call performs the same task as dup(), but instead of using the lowest-numbered unused file descriptor, it uses the descriptor number specified in newfd. If the descriptor newfd was previously open, it is silently closed before being reused.

- The steps of closing and reusing the file descriptor newfd are performed atomically.

Yes both functions are necessary.
dup() can be used if we want to use a new file descriptor with its old file descriptor. That means it allows us to use the both of them interchangeably.
dup2() allows to replace an existing file descriptor with a new one.

ii) In the given code the standard output/input is closed in a separate line before calling the dup() function. In this case a race condition can occur.
When this happen another process can use that opportunity and use it to another different file descriptor other than the file descriptor what we actually want.
(But there were no any output change in my case. It happened as it supposed to happen.)

iii) ex32b.c

c) exercise3.2.c_skel.c

Exercise 4.1

a) A → When mkfifo(fifo,0666); is commented in reader → we have to first run the writer program and then reader in order to show the read message from writer. Otherwise reader do not wait for writer. It just end the program.

B → When mkfifo(fifo,0666); is commented in writer → we have to first run the reader program. It executes and wait for the writer to send the message and display it and end the program. If we run the writer program, the program will just ended. So the message is not send to the reader.

mkfifo() is the function that makes a special file in the specified path of the file system. Reader and writer use that file to read and write. In order to do that it should be called in both ends. So that this function makes read() and write() functions block until the open() is called in other process.

If the makfifo() is omitted error cannot be seen all the time because it is a race condition.

b) ex4.1reader.c
ex4.1writer.c