ASSIGNMENT 1-Part1
Process creation and termination system calls

Ayush Mahant
2019353
CSSS

Program description-

First I have made a first.c file in which I have included the following header files-
1.stdio.h
2.stdlib.h
3.unistd.h
4.sys/types.h
5.sys/stat.h
6.sys/wait.h
7. fcntl.h

In the main,
First using “fork” system call , we receive the parent and child processes.
The total number of bytes in the file is 6785 which I calculated. Using open() I opened the given file “file.csv” in read only mode.
I used “lseek” to start the file status from the second line which is the 79th byte.
I check for if the file is received correctly, which is taken by in line number 23, i.e , if file1 !=-1.
After that, I check if the fork system call is working correcting and not reporting any errors.
If an error is there, I will report it.
Now we go to the child process, here we need to report for only section A. I used “read” system calls for reading the bytes all together from the file from the 79th byte and stores it in buf. If it is read correctly then, I continue,else I report the error. Now in a string 2D array I am storing line by line using strtok which has the delimiter as “\n”. I have initialized the array within which the average will be stored as double. Now, I run the loop for line by line interpretation where each line is separated by a ‘,’ .So I use strtok with the delimiter as ‘,’ . I check for if the element at the second index is ‘B’ or ‘A’ if it is B I use break and get out of the loop for the specific line. Now I add sum for the assignments of each student using “atoi”.
After adding the sum, I find the average and store it in the specific index of the average array.

Now for displaying the file information I have used fgets to get line by line interpretations.
I open the file using “fopen”. I then use fgets for each line and print it on the screen. Along with it I print the average of the current student.I close the file using “fclose”.

I exit the child process using “exit” syscall.
Then it enters the parent process where using the “waitpid” system call , we wait for the child process to finish executing.
Now we go to the parent process, here we need to report for only section B. I used “read” system calls for reading the bytes all together from the file from the 79th byte and stores it in buf. If it is read correctly then, I continue,else I report the error. Now in a string 2D array I am storing line by line using strtok which has the delimiter as “\n”. I have initialized the array within which the average will be stored as double. Now, I run the loop for line by line interpretation where each line is separated by a ‘,’ .So I use strtok with the delimiter as ‘,’ . I check for if the element at the second index is ‘B’ or ‘A’ if it is A, I use break and get out of the loop for the specific line. Now I add sum for the assignments of each student using “atoi”.
After adding the sum, I find the average and store it in the specific index of the average array.

Now for displaying the file information I have used fgets to get line by line interpretations.
I open the file using “fopen”. I then use fgets for each line and print it on the screen. Along with it I print the average of the current student.I close the file using “fclose”.

System calls used with description-

Fork -
Used to create the child process which runs concurrent to the parent process. It returns the integer value.It returns a negative value if the child process is not created. It returns 0 for the newly created child process otherwise greater than 0 for the parent process. The child ID is unique to the parent process id otherwise they are almost the same.
No arguments passed in fork

lseek-
Lseek system call is used to change the position of the filer pointer for read and write. We give the number of bytes from where the reading or writing needs to start. Offset is set to offset bytes using SEEK_SET. It returns the offset of working otherwise -1.
The file name ,the offset, and SEEK_SET is passed as arguments.

read-

```
ssize_t read(int fd, void *buf, size_t count);
```

Read system call, stores from fd to buf the characters upto the number of bytes that is the count.
If count is 0 it returns 0 and such for >0 otherwise it returns -1.
The filename , the storage(string array) and the number of bytes is passes as arguments.

waitpid-

```
pid_t waitpid(pid_t pid, int *status, int options);
```

For a waitpid system call, we need to include “sys/stat.h” and “sys/wait.h” header files.

For waitpid ,it means to wait for any child process whose process group ID is equal to that of the calling process for which the option is 0.
The process id , and the status and 0 is passes as arguments.

Exit -
Exit system call terminates the process normally. Generally 0 status indicates that it is terminating successfully. Status should not be beyond 256.
0 is what is passed.

Open -

```
int open(const char *pathname, int flags);
```

Open system calls open the file represented by the pathname, and returns -1 if not found otherwise >0 value.

Strtok(not a system call) -

```
char *strtok(char *str, const char *delim)
```

Strtok breaks the string “str” into tokens by the delimiter represented by delim.
NULL pointer is returned if no token is to be retrieved.

ERROR HANDLED-

1. If the file does not exist it returns -1, so that error is reported.
2. If there is nothing to be read from the file, i.e when it returns -1, and error is shown.
3. These handlings both take place in the parent and child processes.
4. The other error is when the child process is not created when the fork system call returns a <0 value.
