دانشگاه علم و صنعت ایران
دانشکده کامپیوتر

سیستم‌های عامل

آشنایی با پردازه:
تمرین کلاسی جلسه دوم حل تمرین عملی

سوال اول

در این تمرین قرار است با استفاده از فراخوانی‌های سیستمی fork و خانواده exec برنامه‌ای به زبان C بنویسید که یک برنامه را اجرا کند و خروجی آن برنامه را در یک فایل مشخص ذخیره کند.

مراحلی که در پیاده سازی برنامه باید به آن توجه کنید به این شرح است. در ابتدا ورودی‌های برنامه را دریافت کنید و صحت آن‌ها را بررسی کنید. در صورت وجود مشکل با پیام مناسب برنامه را خاتمه دهید. در غیر اینصورت یک پردازه جدید ایجاد کنید. یک فایل جدید ایجاد کنید. اگر فایل از قبل وجود داشت، مطمئن شوید که محتوای آن از نو نوشته می‌شود. خروجی‌های پردازه را به فایل مورد نظر نگاشت دهید. هم stdout و هم stderr را در فایل خروجی ذخیره کنید. پردازه پدر باید منتظر بماند تا پردازه‌ای که ایجاد کرده است خاتمه پیدا کند. کد خاتمه پردازه ایجاد شده را چاپ کنید.

برای پیاده سازی برنامه به شرایطی که در ادامه می‌آید توجه کنید.

۱. نام برنامه را redirect قرار دهید.

۲. برنامه آدرس یک فایل اجرایی را از کاربر دریافت می‌کند

۳. برنامه باید صحت آدرس فایل اجرایی را بررسی کند. فایل گفته شده در آدرس باید وجود داشته باشد (باید فایل باشد و نه یک directory) و فایل باید قابلیت اجرا داشته باشد.

۴. برنامه بعد از دریافت فایل اجرایی، آدرس یک فایل را دریافت می‌کند. فایلی که آدرس آن داده شده است باید ایجاد شود. درصورتی که فایل از قبل وجود داشته باشد باید محتوای آن از نو نوشته شود. نیازی به چک کردن صحت آدرس نیست. اگر آدرس فایل گفته شده مشکل داشت، به طور مثال پوشه‌های بالایی آن وجود نداشت، برنامه را با یک کد خطا پایان دهید.

۵. پس از دریافت آدرس فایل، پارامترهایی که کاربر می‌خواهد به عنوان ورودی که به برنامه بدهد را دریافت کنید.

۶. در پایان کار، کد خروج برنامه‌ای که در پردازه جدید اجرا شد را چاپ کنید.

۷. برای کامپایل کردن برنامه خود از یک Makefile استفاده کنید.

شکل ۱ پارامترهایی که برنامه به عنوان ورودی انتظار دارد را نشان می‌دهد.

```
→  01 ./redirect
Usage: ./redirect <program> <file> <program arguments>
* program: path to the program to execute
* file: path to the file to store the output
* program arguments: arguments passed to program
→  01
```

شکل ۱: پارامترهایی که برنامه به عنوان ورودی نیاز دارد

شکل ۲ اجرای برنامه را نشان می‌دهد. در دو تصویر اول برنامه به دلیل صحیح نبودن آدرس فایل اجرایی پیام خطا مناسب را نشان داده است. در تصویر سوم، اجرای برنامه نشان داده شده است. آدرس برنامه ls به عنوان فایل اجرایی در اولین پارامتر داده شده است. آدرس یک فایل متنی در دایرکتوری فعلی به عنوان فایل خروجی داده شده است. و در پارامتر سوم مشخص شده است که آدرس پوشه /tmp به عنوان ورود برنامه ls داده شود. همانطور که می‌دانید برنامه ls لیست فایل‌های موجود در یک دایرکتوری را چاپ می‌کند. در تصویر مورد نظر اطلاعات موجود در فایل نیز نمایش داده شده است. همانطور که مشاهده می‌شود لیست فایل‌های موجود در دایرکتوری /tmp در فایل test.txt/. ذخیره شده است.

```
→  01 ./redirect ~/test.txt ./test.txt
Given file is not executable
→  01
```

```
→  01 ./redirect /usr/bin/ls2 ./test.txt
Program path is invalid
→  01
```

```
→  01 ./redirect /usr/bin/ls ./test.txt /tmp/
child rc: 0
→  01 cat test.txt
pamac
sddm-:0-IkWhfC
sddm-auth75e1be10-6644-467f-b371-f2d36e34830f
systemd-private-2a17a601cf844354a2fdfe74980782ff-systemd-logind.service-GJnQTn
systemd-private-2a17a601cf844354a2fdfe74980782ff-systemd-timesyncd.service-tK0h41
systemd-private-2a17a601cf844354a2fdfe74980782ff-upower.service-BrxWM1
vug3TYW
→  01
```

شکل ۲: اجرای برنامه مورد نظر

اطلاعات تکمیلی

هدف از این قسمت یادآوری سریع دستورات و نحوه نگراش فعالیت‌های مرتبط با این تمرین است. در صورت نیاز به آشنایی با این مطالب به جستوجو در باره آن‌ها بپردازید. سوال‌های خود را حتما از گروه حل تمرین بپرسید.

۱.۲ ایجاد یک پردازه جدید

برای اطلاعات بیشتر به توضیحات نوشته شده در manual مراجعه شود. man 2 fork و man 2 wait.

```c
int main()
{
    int pid;
    pid = fork();
    if (pid < 0) {
        printf("creating a new process failed\n");
    } else  if (pid == 0) {
        printf("this is the child process\n");
        return 0;
    }

    // parent continues
    int child_return_code;
    wait(&child_return_code);
    printf("child terminated with rc: %d", child_return_code);
    return 0;
}
```

۲.۲ چگونه یک فایل جدید ایجاد کنیم؟

```c
int main()
{
    char path[] = "/home/user/tmp/test.txt"
    int fd;

    /*
     * int open(const char *pathname, int flags, mode_t mode);
    * */
    fd = open(output_file, O_CREAT | O_WRONLY | O_TRUNC, S_IRUSR | S_IWUSR);
    //...
    return 0;
}
```

Question 2

A web server is an application which can handle requests over the web. One of the characteristics that a web server should have is capable of responding to many clients within the low latency. E.g. consider Google as a web server which millions of clients can get their response simultaneously. A very simple web server program is attached to this homework. This web server only could response to clients one by one. Modify this program to address this issue.

In this question, you should modify the attached file in `q1` directory. Please Include the modified file in your submission with this exact name and directory.

- Step 1: Compile and Run the `server.c`
- Step 2: In your browser: http://localhost:8090/
- Step 3: Add `fork` to `server.c`; Then goto step 1.
- If you get the error: `In bind: Address already in use` try to change PORT number in the file and then recompile and run.

Question 3

In `q2` directory, there is another C code just like the question 1. Use `pthread` library in order to implement a multi-threaded server instead of multi-process server.

- Step 1: Compile and Run the `server.c`
- Step 2: In your browser: http://localhost:8090/
- Step 3: Include `pthread` library and add necessary changes to `server.c`; Then goto step 1.
- If you get the error: `In bind: Address already in use` try to change PORT number in the file and then recompile and run.

Question 4

With completion of the question 1 and 2 you should be able to differentiate the multi-process and multi-thread programming. Write a report and submit it in **PDF** format, using the questions 1 and 2, try to explain what is different between these two approach, which are the advantages and disadvantages of each solution. Also, try to understand how variables look like in memory.