



INTER-PROCESS COMMUNICATION

OPERATING SYSTEM



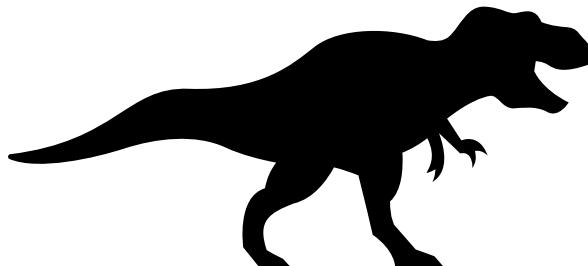
COLLABORATORS NAME & ID :
MARYAM ALIKARAMI 9731045
MAHAN AHMADVAND 9731071



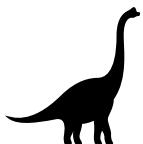


INTER-PROCESS COMMUNICATION

OPERATING SYSTEM



COLLABORATORS NAME & ID :
MARYAM ALIKARAMI 9731045
MAHAN AHMADVAND 9731071



OPERATING SYSTEM LABORATORY

INTER-PROCESS COMMUNICATION



INSTRUCTOR :
ARMAN GHEISARI

COLLABORATORS :
MARYAM ALIKARAMI
MAHAN AHMADVAND

QUESTION 1

محیطی آماده کنید که دو فرآیند در آن وجود داشته باشند و از روش حافظه مشترک برای ارتباط استفاده کنند.(برای مثال می توانید دو فرآیند در نظر بگیرید که کی مقداری بنویسد و دیگر از آن بخواند)
کد مربوط به writer :

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/ipc.h>
4 #include <sys/shm.h>
5 #include <sys/stat.h>
6
7 int main (int argc, char *argv[])
8 {
9     int segment_id;
10    key_t key;
11    key = 3232;
12    char* shared_memory;
13    int shared_segment_size = 100;
14
15    struct shmid_ds shmbuffer;
16    int segment_size;
17
18
19    segment_id = shmget(key, shared_segment_size, IPC_CREAT | 0666);
20
21    shared_memory = shmat (segment_id, NULL, 0);
22    printf("shared memory attached at address %p\n", shared_memory);
23
24    shmctl(segment_id, IPC_STAT, &shmbuffer);
25    segment_size = shmbuffer.shm_segsz;
26    printf("segment size: %d\n", segment_size);
27
28    sprintf(shared_memory, "%s", argv[1]);
29    printf("This string inserted in shared memory: %s\n\n", argv[1]);
30
31    printf("Waiting for reader...\n");
32    while (*shared_memory != '*')
33        sleep(1);
34
35    shmdt(shared_memory);
36
37    return 0;
38 }
39 }
```

ک مریبوط پے : reader

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/stat.h>

int main()
{
    int segment_id;
    key_t key;
    char * shared_memory;
    int shared_segment_size = 100;

    key = 3232;

    segment_id = shmget(key, shared_segment_size, IPC_CREAT | 0666);

    shared_memory = shmat(segment_id, NULL, 0);

    printf("%s\n", shared_memory);

    *shared_memory = '*';

    exit(0);
}

exit(0):
    *shared_memory = '*';

    btrnft("n/segs", "shared-memory");
    shmctl(segment_id, SHM_UNRM, 0);
```

ابتدا می خواهیم توضیحاتی در مورد نحوه ای عملکرد کد های بالا بدهیم سپس خروجی موردنظر را مشاهده کنیم.

در این قسمت از ساختار shared memory برای ارتباط بین process ها استفاده کرده ایم.

برای ایجاد حافظه ای مشترک برای پردازه ها از فراخوانی سیستمی `shmget()` استفاده کرده ایم، پارامتر اول شماره ای segment ای را که می خواهیم اختصاص بدهیم را مشخص می کند، که در اینجا متغیر `key` که مقدار 3232 را برای آن انتخاب کرده ایم به آن اختصاص داده ایم، متغیر بعدی سایز segment ای که می خواهیم به عنوان حافظه ای اشتراکی اختصاص دهیم را مشخص می کند که در اینجا مقدار 100 را به آن اختصاص داده ایم یعنی 100 تا `page` و سپس باید مود دسترسی به این حافظه ای اشتراکی را تعیین کنیم، که در اینجا `IPC_CREAT|0666` را به عنوان ورودی داده ایم، در واقع 0666 برای تعیین دسترسی ها استفاده می شود و `IPC_CREAT` برای این است که به سیستم بگوییم که یک `memory segment` برای حافظه ای اشتراکی ایجاد کند.

خروجی این تابع یک عدد `integer` است.

حال باید این حافظه ای مشترک را به فضای آدرس برنامه را `attach` کنیم، این کار را با استفاده از تابع `(shmat()` انجام می دهیم، که سه تا پارامتر می گیرد، پارامتر اول خروجی `shmget()` است، سپس آدرسی که می خواهیم مپ کنیم را مشخص می کنیم که ورودی `Null` داده ایم که این به این معناست که خود `linux` یک حافظه به ما تخصیص می دهد و پارامتر آخر را نیز صفر داده ایم که این پارامتر تعیین می کند

که این حافظه ای که attach کرده ایم **read only** باشد یا

که مقدار صفر هیچکدام از پارامتر ها را سرت نمی کند.
دستور `(shmctl()` یک دستور کنترلی و اطلاعاتی است که وضعیت حافظه ای اشتراکی را برای ما بازمی گرداند که ورودی اول آن همان خروجی `(shmget()` است و ورودی دوم `IPC_STAT` داده ایم که می گوییم وضعیت حافظه را می خواهیم و ورودی آخر یک داده ساختار است به اسم `shared_memory` که آن را به این دستور پاس داده ایم، حال با استفاده از تابع `sprintf` در حافظه ای اشتراکی می نویسیم و سپس با `while` نوشته شده صبر می کنیم که `reader` این مقدار را از حافظه ای اشتراکی بخواند و زمانی که `reader` از حافظه ای اشتراکی خواند مقدار `*` را در متغیر `shared` ذخیره می کند و `writer` با دریافت این مقدار از `while` بیرون می آید و حافظه ای اشتراکی را با استفاده از دستور `(shmdt()` آزاد می کند.

با استفاده از دستور `gcc -o writer writer.c` و `gcc -o reader reader.c` این دو کد را کامپایل کرده و خروجی را مشاهد می کنیم :

```
mahan@mahan-VirtualBox:~/Desktop$ cd Desktop
mahan@mahan-VirtualBox:~/Desktop$ gcc -o writer writer.c
writer.c: In function 'main':
writer.c:33:9: warning: [enabled by default] implicit declaration of function 'sleep'; did you mean 'select'? [-Wimplicit-function-declaration]
    sleep(1);
          ^
          sleep
mahan@mahan-VirtualBox:~/Desktop$ gcc -o reader reader.c
reader.c: In function 'main':
reader.c:26:5: warning: [enabled by default] implicit declaration of function 'exit' [-Wimplicit-function-declaration]
    exit(0);
    ^
reader.c:26:5: warning: incompatible implicit declaration of built-in function 'exit'
reader.c:26:5: note: include <stdlib.h> or provide a declaration of 'exit'
reader.c:11:1: warning: include <stdlib.h>
reader.c:11:1: note: include <stdlib.h>
reader.c:26:5:
    exit(0);
    ^
mahan@mahan-VirtualBox:~/Desktop$ ./writer Hello
shared memory attached at address 0x7f0bbfd10000
segment size: 100
This string inserted in shared memory: Hello
Waiting for reader...
mahan@mahan-VirtualBox:~/Desktop$
```

همانطور که مشاهد می کنیم خروجی مورد نظر را دریافت کرده ایم.

QUESTION 3

محیطی را فراهم کنید که در آن دو فرآیند با استفاده از خط لوله به تبادل یک پیام متنی پردازنند. فرآیند اول یک پیام متنی دارای حروف بزرگ و کوچک(برای مثال : This Is First (Process) به فرآیند دوم ارسال می کند، فرآیند دوم این پیام را دریافت می کند و حروف بزرگ را به حروف کوچک و حروف کوچک را به حروف بزرگ تبدیل می کند(برای مثال : iS fIRST (process) و به فرآیند اول می فرستد. راهنمایی : برای این کار به دو خط لوله نیاز دارید.

کد مربوط به این سوال مطابق شکل زیر است :

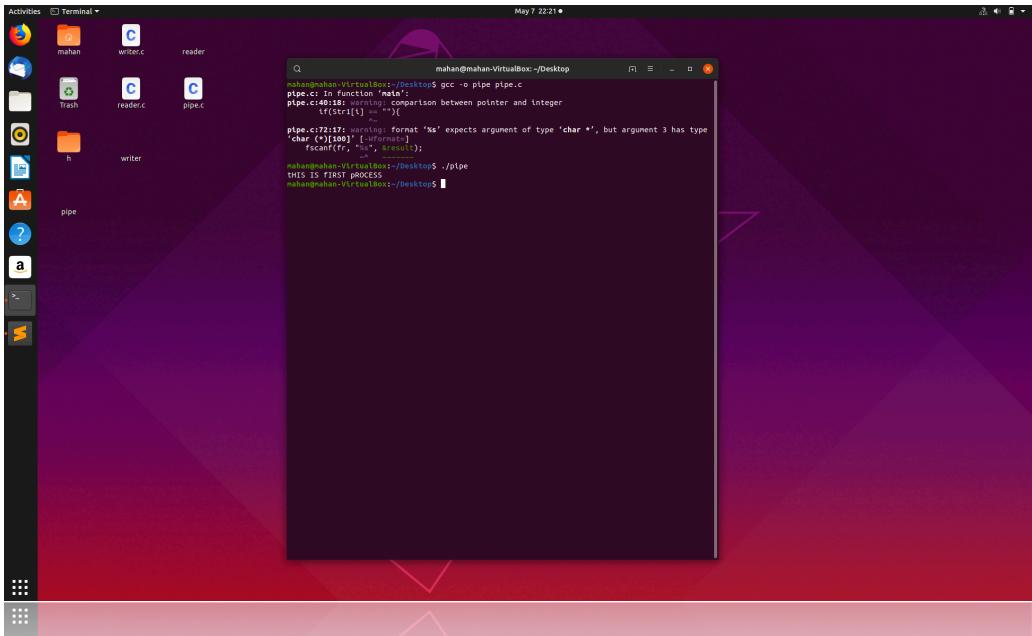
حال به توضیح قسمت های مختلف این کد می پردازیم :
ابتدا در خط اول file descriptor های پایپ را مشخص کرده ایم سپس یک آرایه ای از کاراکترها داریم که مقادیر آن را با This is First Process پر کرده ایم، سپس در خط 13، Error handling مربوط به پایپ را انجام داده ایم و برای این کار از perror استفاده کرده ایم که ارور را در stderr می نویسد و به آن system pipe() را پاس داده ایم که یک pipe() است که برای ارتباط بین پردازه های مختلف linux function استفاده می شود.

حال با استفاده از دستور fork یک پردازه دیگر(پردازه فرزند) ایجاد کرده ایم که خروجی fork برای فرزند صفر بوده و برای پدر آن یک عدد مثبت که PID فرزند خود است را برمی گرداند، حال با استفاده از دستور switch case، کارهای پدر و فرزند را از هم تفکیک می کنیم.(که حالت default را برای پدر انتخاب کرده ایم و حالت صفر برای فرزند است)

سپس مطابق کد عمل تبدیل lowercase ها به uppercase ها و بالعکس مشخص است(توجه شود که این کار توسط فرزند انجام می شود).

سپس حاصل نهایی را برای پدر می فرستیم و آن را در صفحه ی ترمینال چاپ می کنیم.

حال می خواهیم برنامه ی موردنظر را کامپایل کرده و خروجی را مشاهده کنیم.



همانطور که مشاهده می کنیم خروجی به درستی نمایش داده شده است.

THIS IS FIRST pROCESS