گزارش آزمایش ۵

فاطمه باقری ۹۵۱۰۵۴۱۹

حسین قطب‌الدینی ۹۵۱۰۹۹۷۲

اسرا کاشانی‌نیا ۹۵۱۰۵۸۱۶

الف)

pipe () یک pipe ایجاد می کند ، یک کانال داده یک طرفه است که می تواند برای ارتباط بین پردازه‌ای استفاده شود.آرایه pipefd برای برگرداندن دو اشاره‌گر به دو فایل به pipe استفاده می شود. 0] pipefd] به انتهای read لوله اشاره دارد. 1] pipefd] به انتهای write لوله اشاره دارد. داده های نوشته شده در انتهای write لوله توسط هسته بافر می شود تا زمانی که از انتهای read لوله خوانده شود.

ب) می‌دانیم که ID پردازه‌ی فرزند ۱ است و ID پردازه‌ی والد یک عدد مثبت است.

```cpp
#include <iostream>
#include <unistd.h>
#include <cstdio>
#include <cstring>

using namespace std;

int main()
{
    int fd[2];
    int sth = pipe(fd);

    pid_t pid = fork();

    if (pid < 0) {
        cerr << "Fork was not successful" << endl;
        return 1;
    }
    else if (pid == 0){
        close(fd[1]);
        char buffer[100];
        read(fd[0], buffer, 99);
        cout << "Read " << buffer << " from pipe." << endl;
    }
    else{
        close(fd[0]);
        char* str = "Hello world";
        write(fd[1], str, strlen(str) + 1);
    }

    return 0;
}
```

C++ Tab W

خروجی:

```
esra@esra-HP-Spectre-Notebook:~/Desktop$ g++ forkpipe.cpp
forkpipe.cpp: In function ‘int main()’:
forkpipe.cpp:27:21: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
         char* str = "Hello world";
                     ^~~~~~~~~~~~~
esra@esra-HP-Spectre-Notebook:~/Desktop$ ./a.out
Read Hello world from pipe.
esra@esra-HP-Spectre-Notebook:~/Desktop$
```

فعالیت: می‌دانیم که کد ورودی استاندارد ۰ و کد خروجی استاندارد ۱ است. پس در پردازنده‌ی فرزند با دستور dup2(fd[0], 0) **کاری می‌کنیم که ورودی به جای** stdin **از** fd[0] **(انتهای** read **پایپ) خوانده شود و در پردازه‌ی والد با دستور** dup2(fd[1], 1) **کاری می‌کنیم که خروجی دستور نوشتن به جای خروجی استاندارد روی** fd[1] **(طرف** right **پایپ) نوشته شود.**

```cpp
#include <unistd.h>
#include <cstdio>

using namespace std;

int main()
{
    int fd[2];
    int sth = pipe(fd);
    pid_t pid = fork();

    if (pid < 0) {
        fprintf(stderr, "Fork was not successful");
        return 1;
    }
    else if (pid == 0){
        close(fd[1]);
        dup2(fd[0], 0);
        execl("/usr/bin/wc", "/usr/bin/wc", NULL);
    }
    else{
        close(fd[0]);
        dup2(fd[1], 1);
        execl("/usr/bin/ls", "/usr/bin/ls", NULL);
    }

    return 0;
}
```

خروجی:

```
esra@esra-HP-Spectre-Notebook:~/Desktop$ ./a.out
esra@esra-HP-Spectre-Notebook:~/Desktop$       0       0       0

esra@esra-HP-Spectre-Notebook:~/Desktop$
```

حال این یک pipe یک‌طرفه (half duplex) است. اگر یکی دیگر هم بسازیم فقط کدهای بلوک else if و بلوک else در برنامه‌ی بالا را جابجا کنیم، یک half duplex در جهت مخالف خواهیم داشت. با داشتن دو half duplex در جهت‌های مخالف ارتباط تمام‌دوطرفه خواهیم داشت.

۲) سیگنال‌ها

الف)

کد ۴: SIGILL: دستور غیرمعتبر (illegal). یعنی کد پردازه حاوی دستور زبان ماشینی بوده است که پردازنده نمی‌تواند آن را بفهمد.

کد ۱۷ (SIGCHLD): یعنی پردازه قبلا با دستور fork یک یا چند پردازه‌ی فرزند به وجود آورده که الان یکی از آنها تمام شده است (یا به طور طبیعی یا kill)

کد ۱۹ (SIGSTOP): یعنی پردازه توسط سیستم عامل متوقف می‌شود به این صورت که وضعیت (ثبات‌ها و..) آن ذخیره می‌شود اما به آن چرخه‌های CPU اختصاص داده نمی‌شود.

کد ۲۹ (SIGIO): پردازه می‌تواند مقدر کند که در یکی از این دو حالت سیگنال SIGIO برای آن ارسال شود: یا اینکه داده‌ی ورودی برای پردازش شدن توسط پردازه آماده شود، یا اینکه کانال خروجی برای اینکه پردازه روی آن داده‌ای بنویسد آماده شود.

کد ۱۱ (SIGSEGV) یعنی پردازه تلاش کرده از آدرسی داده بخواند که برای آن پردازه اختصاص نیافته بوده است (مثلا در حالاتی که می‌خواهیم عنصر ۱۰۱ ام یک آرایه‌ی ۱۰۰ عضوی را بخوانیم).[1]

ب)

[1] https://www-uxsup.csx.cam.ac.uk/courses/moved.Building/signals.pdf

alarm () ترتیبی می دهد تا یک سیگنال SIGALRM در چند ثانیه به پردازه‌ای که دارد اجرا می‌شود و آن را فراخوانی کرده تحویل داده شود. اگر seconds صفر باشد ، هرگونه alarm در انتظار لغو می شود. در هر صورت هرگونه alarm از قبل تنظیم شده لغو می شود.

alarm () تعداد ثانیه های باقیمانده را تا زمان تحویل alarm تنظیم شده قبلی برمی‌گرداند ، یا اگر alarm برنامه ریزی شده قبلی وجود ندارد ، مقداری که تابع alarm برمی‌گرداند صفر است.

ج)

با اجرای برنامه، اول Looping forever . . . چاپ می‌شود، ۵ ثانیه بعد Alarm clock چاپ می‌شود و برنامه تمام می‌شود. دلیلش این است که رفتار پیش‌فرض برنامه این است که هرگاه ثانیه‌های alarm تمام شود، پردازه هم kill می شود، چه تا آن زمان یک درصدش اجرا شده باشد چه ۹۹ درصد آن. پس چون اجرای while(1) قطعا تا ۵ ثانیه بعد از شروع اجرا تمام نمی‌شود، هر چیزی که بعد از آن باشد هم هرگز اجزا نمی‌شود.

```
esra@esra-HP-Spectre-Notebook:~/Desktop$ sudo g++ alarms.cpp
esra@esra-HP-Spectre-Notebook:~/Desktop$ ./a.out
Looping forever ...
Alarm clock
esra@esra-HP-Spectre-Notebook:~/Desktop$
```

د)

نکته این است که با اجرای خط signal(SIGALRM, endloop) هرگاه alarm به آخر برسد و سیگنال SIGALARM را تولید کند، به جای رفتار عادی برنامه که kill شدن است، تابع endloop فراخوانی می‌شود.

```cpp
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

using namespace std;

bool flag = false;

void endloop(int i){
    flag = true;
}

int main()
{
    alarm(5);
    printf("Looping forever ... \n");

    signal(SIGALRM, endloop);
    while(!flag){

    }
    printf("This line should never be printed");
    return 0;
}
```

خروجی کد:

```
esra@esra-HP-Spectre-Notebook:~/Desktop$ g++ endloop.cpp
esra@esra-HP-Spectre-Notebook:~/Desktop$ ./a.out
Looping forever ...
This line should never be printedesra@esra-HP-Spectre-Notebook:~/Desktop$
```

فعالیت:

تعداد ctrl+c ها را در ثبات counter نگه می‌داریم.

```cpp
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

using namespace std;

int counter = 0;

void press(int);

void addsignal(int signl){
    if(signal(SIGINT, press) == SIG_ERR){
        printf("Error on signal %d\n", signl);
        return;
    }
}

void press(int num){
    counter++;
    printf("Ctrl + %c signal number %d \n", num==SIGINT? 'C' : 'Z', counter);
    addsignal(num);
    if(counter >= 2){
        exit(num);
    }
    return;
}

int main()
{
    printf("start\n");
    addsignal(SIGINT);
    addsignal(SIGSTOP);
    while(true){
        pause();
    }
    return 0;
}
```

خروجی:

```
esra@esra-HP-Spectre-Notebook:~/Desktop$ g++ ctrlc.cpp
esra@esra-HP-Spectre-Notebook:~/Desktop$ ./a.out
start
^CCtrl + C signal number 1
^CCtrl + C signal number 2
esra@esra-HP-Spectre-Notebook:~/Desktop$
```