



سؤال ۱. ایجاد pipe یک‌سویه

۱. برای ساخت ارتباط یک‌سویه از دستور و فراخوانی سیستمی pipe استفاده می‌شود.

```
man 2 pipe

File Edit View Search Terminal Help
PIPE(2) Linux Programmer's Manual PIPE(2)

NAME
    pipe, pipe2 - create pipe

SYNOPSIS
    #include <unistd.h>

    int pipe(int pipefd[2]);

    #define _GNU_SOURCE          /* See feature_test_macros(7) */
    #include <fcntl.h>          /* Obtain O_* constant definitions */
    #include <unistd.h>

    int pipe2(int pipefd[2], int flags);

DESCRIPTION
    pipe() creates a pipe, a unidirectional data channel that can be used
    for interprocess communication. The array pipefd is used to return two
    file descriptors referring to the ends of the pipe. pipefd[0] refers
    to the read end of the pipe. pipefd[1] refers to the write end of the
    pipe. Data written to the write end of the pipe is buffered by the
    kernel until it is read from the read end of the pipe. For further
    Manual page pipe(2) line 1 (press h for help or q to quit)
```

شکل ۱: عمل‌کرد pipe

۲. همان‌طور که در مستند جلسه‌ی پنجم توضیح داده شده است، برای ساخت یک pipe جدید کافی است به‌صورت زیر عمل کنیم:

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

int main()
{
    int pipefd[2];
    int result = pipe(pipefd);
    if (result == 0)
    {
```

```

    printf("pipe done");
}
return 0;
}

```

۳. برای انتقال پیام Hello world! از پدر به فرزند و چاپ آن در خروجی باید کد زیر را اجرا کنیم:

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define MSG_SIZE 11

int main()
{
    int pipefd[2];
    int result = pipe(pipefd);
    if (fork() > 0)
    {
        printf("parent is executed...\n");
        close(pipefd[0]);
        printf("writing message in the file descriptor...\n");
        write(pipefd[1], "Hello world", MSG_SIZE);
        printf("closing read file descriptor in parent...\n");
        close(pipefd[1]);
    }
    else {
        char msg[MSG_SIZE];
        printf("child is executed...\n");
        read(pipefd[0], msg, MSG_SIZE);
        printf("%s", msg);
    }
    return 0;
}

```

۴. در این قسمت می‌خواهیم خروجی پردازشی پدر (دستور ls) را به عنوان ورودی به پردازشی فرزند بدهیم.

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int pipefd[2];
    pipe(pipefd);

    if (fork() > 0)
    {
        close(pipefd[0]);
        dup2(pipefd[1], STDOUT_FILENO); //make output go to the pipe
        execlp("ls", "ls", "-a", (char *) NULL);
    }
}

```

```

    close(pipefd[1]);
    dup2(pipefd[0], STDIN_FILENO); //get input from pipe
    execlp("wc", "wc", (char *) NULL);
}

```

۵. برای ایجاد ارتباطات تمام دو طرفه بین پردازنده‌ها باید دو بار از pipe استفاده کرد. به کمک یکی از آن‌ها در یکی بخوانیم و به کمک دیگری بنویسیم.

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main(void)
{
    int pid, n, c, p, k, nbread;
    char buf1[12], buf2[12];
    int fd1[2], fd2[2];
    pipe(fd1);
    pipe(fd2);
    pid = fork();
    if (pid == 0)
    {
        close(fd1[1]);
        close(fd2[0]);
        read(fd1[0], buf2, sizeof(buf2));
        n = atoi(buf2);
        printf("Child read %d\n", n);
        for (int i = 0; i < n; i++)
        {
            printf("child dozes...\n");
            sleep(3);
            printf("child wakes...\n");
            nbread = read(fd1[0], buf2, sizeof(buf2));
            if (nbread == (1-
            {
                fprintf(stderr, "child exits after read failure\n");
                exit(1);
            }
            c = atoi(buf2);
            c = c * 2;
            sprintf(buf2, "%d", c);
            write(fd2[1], buf2, sizeof(buf2));
            printf("Child wrote [%s]\n", buf2);
        }
        close(fd1[0]);
        close(fd2[1]);
        printf("Child done\n");
        exit(0);
    }
    else

```

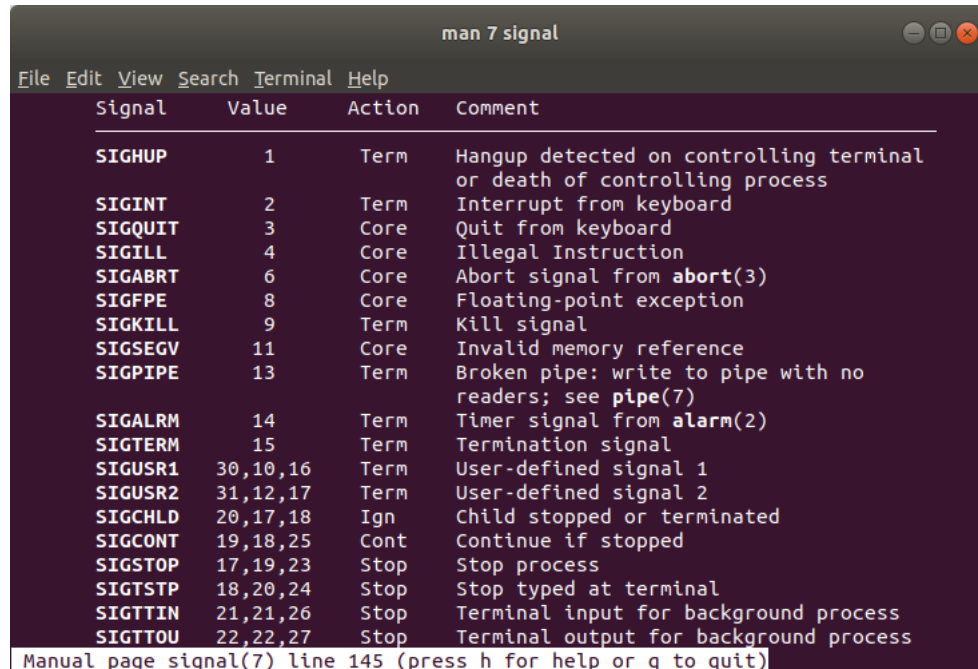
```

{
    close(fd1[0]);
    close(fd2[1]);
    printf("Enter integer: ");
    scanf("%d", &p);
    sprintf(buf1, "%d", p);
    write(fd1[1], buf1, sizeof(buf1));
    printf("Parent wrote [%s]\n", buf1);
    printf("parent dozes...\n");
    sleep(3);
    printf("parent wakes...\n");
    for (int i = 0; i < p; i++)
    {
        sprintf(buf1, "%d", i);
        write(fd1[1], buf1, sizeof(buf1));
        printf("parent wrote [%s]\n", buf1);
        read(fd2[0], buf2, sizeof(buf2));
        printf("number is: %s\n", buf2);
    }
    close(fd1[1]);
    close(fd2[0]);
    wait(NULL);
}
return 0;
}

```

سؤال ۲. سیگنال‌ها

۱. در این قسمت با استفاده از دستور `man 7 signal` اطلاعات راجع به این فراخوان سیستمی را می‌توانیم مشاهده کنیم.



Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from <code>abort(3)</code>
SIGFPE	8	Core	Floating-point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers; see <code>pipe(7)</code>
SIGALRM	14	Term	Timer signal from <code>alarm(2)</code>
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

Manual page signal(7) line 145 (press h for help or q to quit)

شکل ۲: لیستی از signal‌ها

- **SIGINT**: متوقف شدن برنامه از طریق کیبورد را تشخیص می‌دهد.
- **SIGQUIT**: از طریق کیبورد از پردازش خارج می‌شود.
- **SIGTERM**: سیگنالی است که بیان‌کننده‌ی خاتمه یک پردازش است.
- ... و ...

۲. از `alarm()` برای ارسال `SIGALRM` به پردازش استفاده می‌شود. آرگومان اصلی آن زمان است که واحد آن به ثانیه است. اگر مقدار آن صفر باشد، `alarm` خاصی برنامه‌ریزی نمی‌شود.

۳. کد زیر با توجه به `alarm(5)` ۵ ثانیه اجرا می‌شود و چون داخل حلقه‌ی بی‌نهایت است، بنابراین دیگر خط‌ها (همان‌طور که در `printf` هم مشاهده می‌شود) اجرا نمی‌شوند و برنامه خاتمه پیدا می‌کند.

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    alarm(5);
    printf("Looping forever...\n");
    while(1);
    printf("This line should never be executed.\n");
    return 0;
}
```

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

int flag = 1; // global flag

void alarm_handler(int signum) {
    flag = 0;
}

int main()
{
    signal(SIGALRM, alarm_handler); // Register signal handler
    alarm(5);
    printf("Looping forever...\n");
    while(flag){
        pause();
    };
    printf("This line should be executed.\n");
    return 0;
}
```

۵. خروج از برنامه پس از دو بار فشردن کلید ترکیبی CTRL + C

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

int counter = 1; // global flag

void handler(int signum) {
    if (counter == 2){
        printf("\nexiting...\n");
        exit(1);
    }
    else {
        printf("\npres CTRL + C again to exit.\n");
        counter += 1;
    }
}

int main()
{
    signal(SIGINT, handler);
    while(1);
}
```