دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

آزمایشگاه سیستم عامل

علی فانیان

زینب زالی

تابستان ۱۳۹۸

مروری بر روش‌های مبتنی بر پیام ارتباط میان فرآیندها (Pipe-Socket-Signal)

Ordinary Pipe

- Ordinary pipe یا مختصرا pipe جهت برقراری ارتباط بین پروسس‌ها، عملکردی شبیه به File دارد با این تفاوت که Pipe در حافظه ی RAM قرار می گیرد ولی File بر روی حافظه‌ی دائمی قرار می گیرد، از این رو Pipe از File سریعتر عمل می‌کند.
- با استفاده از pipe برقراری ارتباط بین فرآیند والد و فرآیند فرزند و همچنین برقراری ارتباط بین فرآیندهای فرزند با یکدیگر امکان‌پذیر است.
- در هر دو حالت ذکر شده Pipe ها، توسط والد ایجاد می‌شوند.
- Pipe در مقایسه با socket از سرعتی بیشتر برخوردار است، در عین حال قابلیت‌های کنترلی کمتری دارد.
- مدیریت pipe ها به دلیل آنکه همواره بایستی توسط یک فرآیند والد ایجاد شود پیچیدگی بیشتری دارد.

Named-Pipe

- Named pipe (خط لوله نام‌گذاری‌شده) عملکردی مشابه pipe دارد با این تفاوت که فرآیندها با داشتن نام آن می‌توانند به آن دسترسی داشته باشند و حتما رابطه‌ی والد-فرزند بین آنها برقرار نیست.
- این خط لوله برای ارتباط بین فرآیندهایی که لزوما رابطه‌ی والد-فرزند ندارند مناسب است.

Socket

- Socket معمولاً جهت ایجاد ارتباط بین دو فرآیند با استفاده از IP Address و Port Number به کار می‌رود.
- اغلب هنگامی از این روش استفاده می‌شود که دو فرآیند ارتباط والد/فرزند نداشته باشند و یا دو فرآیند قصد ارتباط روی شبکه را داشته باشند.
- سوکت‌های تعریف شده در POSIX انواع متعددی دارند که از جمله پرکاربردترین آنها سوکت های TCP/UDP/UNIX می باشند.
- نوع سوکت استفاده شده، سرعت و کارایی(performance) آن را تعیین می کند، برای مثال استفاده از سوکت های نوع UDP یا TCP باعث کمتر شدن سرعت ارتباط می شود ولی استفاده از سوکت های نوع UNIX برای دو فرآیند که روی یک ماشین قرار دارند سرعت و کارایی بالایی خواهد داشت (سرعتی قابل قیاس با سرعت PIPE).

Signal

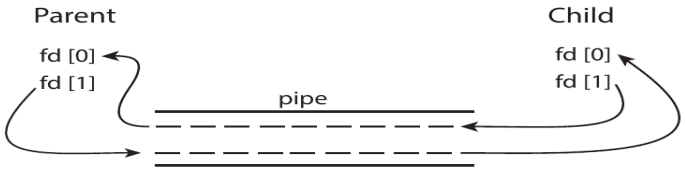
- در این روش، ارتباط بین فرآیندها از راه کنترل رخدادهای خاص و تعریف رفتار فرآیند در قبال رخدادهای تعریف شده است.
- برخلاف socket و pipe نمی‌توان مقداری را از طریق signal منتقل کرد، این روش تنها برای آگاهی از اتفاق افتادن یک رخداد خاص به کار می رود .
- Signal ها نسبت به pipe و socket سرعت کمتر و پیچیدگی‌های بیشتری دارند و به همین دلیل بایستی با احتیاط به کار برده شوند.

در این جلسه آزمایشگاه، با برنامه‌نویسی دو نوع pipe معرفی‌شده آشنا می‌شوید و روش‌های دیگر در جلسات بعدی بیان می‌شوند.

ایجاد ارتباط با Ordinary pipe

در این روش، یک خط لوله ارتباطی بین دو پروسس ایجاد می‌شود. دو پروسس از طریق این خط لوله می‌توانند پیامی ارسال یا دریافت کنند. روش برقراری این نوع ارتباط به شرح زیر است:

- pipe باید توسط یک پروسس والد ایجاد شود.
- با فراخوانی سیستمی pipe(int fd[2]) توسط والد یک خط لوله ایجاد می‌شود. این فراخوانی سیستمی آرگومانی از نوع یک آرایه دوعضوی int دریافت می‌کند.
- پروسس‌هایی که بعد از فراخوانی pipe توسط fork ساخته می‌شوند و همچنین خود پروسس والد می‌توانند از pipe استفاده کند.
- اگر اجرای pipe موفقیت‌آمیز باشد، آرایه fd حاوی دو descriptor خواهد بود که جهت ارسال و دریافت پیام استفاده می‌شود. fd[1] برای هر پروسسی که از آن استفاده کند جهت ارسال پیام به پروسس دیگر و fd[0] جهت دریافت پیام از پروسس دیگر استفاده می‌شود. به عبارت دیگر هر یک از دو سر pipe در هر پروسس مانند شکل، جهت ارسال یا دریافت پیام استفاده می‌شود.
- جهت ارسال و دریافت پیام توابع write و read به همان صورتی که در فایل‌ها استفاده می‌شد، فراخوانی می‌شوند. بنابراین پروسس ارسال‌کننده از write روی fd[1] و پروسس دریافت‌کننده از read روی fd[0] استفاده می‌کند.
- با فراخوانی تابع close(int fd) روی هر یک از descriptorها، یک سر pipe بسته می‌شود. برای مثال close(fd[1]) در هر پروسس سر ارسال پیام روی آن pipe را می‌بندد.
- روال منطقی در استفاده از pipe این است که به صورت یک‌طرفه استفاده شود، بدین منظور هر پروسس باید فقط جهت ارسال یا جهت دریافت پیام از یک pipe استفاده کند و همزمان از یک pipe جهت ارسال و دریافت استفاده نکند. لذا در هر پروسس، هر سر pipe که با آن کار ندارد باید بسته شود. مثلاً اگر پروسسی قصد دارد روی یک pipe پیام بفرستد باید fd[0] را که برای دریافت پیام است close کند.
- در صورتی که از pipe به صورتی که در بند قبل بیان شد استفاده شود، در صورتی که پروسس فرستنده تمام شود یا سر ارسال را close کند، پروسسی که در حال خواندن از pipe بوده است با خروجی منفی تابع read مواجه می‌شود در غیر این صورت تابع read متوجه بسته‌شدن pipe نمی‌شود و برای دریافت یک پیام بلاک می‌شود.



مثال pipe

در این قسمت یه برنامه نمونه pipe نشان داده شده‌است. این برنامه را در یک فایل c کپی کرده و کامپایل و اجرا کنید و عملکرد آن را مشاهده نمایید. در این برنامه یک pipe بین والد و فرزندش استفاده می‌شود.

پس از مشاهده اجرای برنامه خطی، خط close(fd[1]) که در برنامه با ** مشخص شده‌است، کامنت کرده و دوباره برنامه را کامپایـل و اجرا کنید. چه تفاوتی در اجرا می‌بینید؟

```
/*
making a pipe between parent and child, sending messages from parent to child
*/

#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

int main(){
	int fd[2];
	char buffer[256];
	int x=pipe(fd);
	printf("fd=%d\n",fd[0]);
	pid_t pid;
	pid=fork();
	if(pid==0) //in child
	{
		close(fd[1]); //**comment this line
		while(1)
		{
			printf("in child\n");
			if(read(fd[0],buffer,255)>0)
				printf("%s\n",buffer);
			else
				printf("pipe is not available\n");
		}
	}
	else// in parent
	{
		close(fd[0]);
		int i = 0;
		while(i<10)
		{
			printf("in parent\n");
			sprintf(buffer,"message %d to child", i++);
			write(fd[1],buffer,255);
			sleep(2);
		}
	}
	return 0;
}
```

ایجاد ارتباط با named pipe

با استفاده از named pipe می‌توان خط لوله‌های نامداری ایجاد کرد. کار با خط لوله نامدار خیلی بیشتر از خط لوله معمولی شبیه کار با فایل است. از آن‌جایی که named pipe دارای نام است. بعد از ایجاد چنین pipeی در هر پروسس امکان استفاده از آن با کمک نامش وجود دارد. بنابراین نیاز به پروسس والد جهت ایجاد آن نیست. روش برقراری این نوع ارتباط به شرح زیر است:

- برای ایجاد این نوع خط لوله کافیست از فراخوانی سیستمی mkfifo(const char *pathname, mode_t mode) استفاده شود. Pathname یک نام فایل (مسیر فایل) را مشخص می‌کند. این نام، نامی است که قصد داریم برای named pipe خود انتخاب کنیم. mode هم‌سطح دسترسی به این pipe را شبیه سطح دسترسی فایل‌ها مشخص می‌کند.
- پس از ایجاد یک named pipe، هر پروسس که قصد استفاده از آن را داشته باشد، باید با فراخوانی سیستمی open مانند فایل‌ها آن را باز کند.
- پس از بازکردن named pipe، برای ارسال و دریافت پیام از توابع write و read مانند فایل‌ها استفاده می‌شود.

مثال named pipe

این برنامه جهت ایجاد ارتباط با named pipe بین یک والد و فرزند نوشته شده‌است. برنامه را در یک فایل c کپی و کامپایل و اجرا کنید. سپس در یک ترمینال دیگر، محتویات شاخه برنامه خود را مشاهده کنید. آیا فایلی با نام pipeی که ایجاد کرده‌اید وجود دارد؟ در صورت مثبت‌بودن جواب، فایل مربوطه را با دستور cat مشاهده کنید. چه اتفاقی در ادامه اجرای برنامه c شما می‌افتد؟ سعی کنید آنچه می‌بینید را بر مبنای کار named pipe شرح دهید.

```
/*
making a pipe between parent and child and sending messages from parent to child
*/

#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <sys/stat.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>

int main(){
	int pipe, inChild;
	int tmp;
	char path[20];
	sprintf(path,"1.pipe");
	printf("%s\n",path);

	//making the named-pipe
	mkfifo(path,0777);

	char buffer[256];
	bzero(buffer,256);
	pid_t pid;
	pid=fork();
	inChild=0;
	if (pid==0)
		inChild=1;
	while(inChild==1)
	{
		/*
		  operations on named-pipe are similar to a file
		  we open the named-pipe
		*/
		pipe=open(path,O_RDONLY|O_NONBLOCK);
		read(pipe,buffer,255);
		printf("child <- %s\n",buffer);
		bzero(buffer,256);
		sleep(1);
	}
	tmp=0;
	pipe=open(path,O_WRONLY);
	while(inChild==0)
	{
		sprintf(buffer,"%d",tmp);
		tmp++;
		printf("parent -> %s", buffer);
		write(pipe,buffer,strlen(buffer));
		sleep(1);
	}
	return 0;
}
```