باسمه تعالی

تمرین کامپیوتری شماره‌ی ۲

سیستم عامل – بهار ۱۳۹۲

دانشکده مهندسی کامپیوتر

دانشگاه علم و صنعت

مهلت تحویل: ۱۳۹۲/۰۳/۰۳

همانطور که میدانید برای برقراری protection در سیستم اجازه انجام هر کاری به صورت مستقیم به کاربر داده نمیشود و کاربر در خواست های خود(مثلا عملیات I/O) را از طریق System call ها به اطلاع kernel میرساند تا این عملیات در kernel اجرا شوند. شما در این تمرین قرار است با مفاهیم زیر آشنا شوید:

- System call
- Socket programming
- Client-Server System

صورت مسئله :

شما در این تمرین کامپیوتری قرار است یک سیستم ارتباطی برای کارمندان یک شرکت طراحی کنید. این سیستم از یک سرور و تعدادی کلاینت تشکیل شده است به این صورت که هر کدام از کارمندان به وسیله برنامه کلاینتی که در اختیارشان قرار داده میشود از طریق یک سوکت (socket) به سروری که روی کامپیوتر رئیس شرکت اجرا میشود متصل شده تا سرویس هایی از قبیل انتقال فایل و پیام به یکدیگر را دریافت کنند.

جزئیات پیاده سازی :

برنامه server با دو آرگمان دایرکتوری اولیه و شماره پورت اتصال Initialize میشود:

```
./server  directory_for_server  port_number
```

با اجرای برنامه سرور دایرکتوری ذکر شده به عنوان مسیر پیشفرض سرور ساخته شده و سرور روی پورت ذکر شده منتظر دریافت درخواست ها میشود یا اصطلاحا روی این پورت listen میکند.

Client ها نیز باید به صورت زیر اجرا شوند:

```
./client client_name dir
```

به این ترتیب client ای با نام client_name و مسیر پیشفرض dir برای ذخیره فایل هایی که از شبکه دانلود میکند اجرا میشود. این client در این شرایط به هیچ سروری متصل نیست و تنها میتواند از طریق دستور زیر به سرور های در حال اجرا وصل شود:

```
@connect  server_ip:port_which_server_is_listening_into
```

دقت کنید که کارکتر @ به معنی این است که کاربر در حال نوشتن دستور است.

پس از هر اتصال سرور انتظار دارد که client نام خود را برای او بفرستد و اگر این client ، i امین نفری باشد که به سرور متصل میشود از این پس با نام cli#i:client_name در سرور شناخته میشود. اتصال موفقیت آمیز کلاینت در هر دو ترمینال سرور و کلاینت اعلام میشود.

درخواست های هر client پس از اتصال به سرور به صورت های زیر میباشد:

- **درخواست لیست همکاران متصل به سرور**:

```
@get-clients-list
```

سرور در هر شرایطی (حتی اگر در حال دریافت و ارسال فایل باشد) به این درخواست پاسخ میدهد و client در پنجره خود جواب را به صورت زیر مشاهده میکند:

```
----------currently available clients----------
Cli0:Soghra
Cli1:Ghodrat
Cli2:Akbar
Cli5:Aghdas
```

(لیست بالا نشان دهنده این است که cli3 و cli4 از سرور disconnect شده اند.)

- **درخواست share کردن یک فایل:**

```
@share file_path password
```

در صورتی که سرور در حال دریافت یا ارسال فایلی نباشد به این درخواست جواب مثبت میدهد و فایل مورد نظر از طریق پروتکل زیر از کامپیوتر کلاینت به دایرکتوری پیشفرض در کامپیوتر سرور منتقل خواهد شد.

Client		**Server**
Sharefile request	->	
	<-	If not busy then Acknowledge
If Acknowledge continue		
Going to Send packet	->	
		Waiting
Write packet 0	->	Receive and append
Going to Send Packet	->	
		Waiting
Write packet 1	->	Receive and append
Going to Send Password	->	
		Waiting
Write password		Receive and save

- **درخواست لیست فایل های share شده:**

```
@get-files-list
```

سرور در هر شرایطی (حتی اگر در حال دریافت و ارسال فایل باشد) به این درخواست پاسخ میدهد و client در پنجره خود جواب را به صورت زیر مشاهده میکند:

```
----------shared files-----------
Felan.txt          owner:cli#0:Soghra
Bahman.txt         owner:cli#0:Soghra
Pazhman.txt        owner:cli#5:Akbar
```

- **درخواست دریافت یک فایل share شده:**

```
@get filename password
```

در صورتی که سرور در حال دریافت یا ارسال فایلی نباشد، فایل موجود بوده و password صحیح باشد به این درخواست جواب مثبت میدهد و فایل مورد نظر از طریق پروتکل زیر از کامپیوتر سرور به دایرکتوری پیشفرض در کامپیوتر کلاینت منتقل خواهد شد.

Client		**Server**
get request	->	
	<-	If not busy and file exists and correct password then Acknowledge
If Acknowledge continue		

Give next packet	->	
	<-	Write next packet
If not finished then receive and append		
Give next packet	->	
	<-	Write next packet
If not finished then receive and append		
Give next packet	->	
	<-	Write finish code
End of asking for next packet		

پس از اتمام انتقال فایل درنتیجه ی دو درخواست دریافت و ارسال فایل شما باید ضمن اعلام موفقیت آمیز بودن انتقال در ترمینال سرور و کلاینت متوسط سرعت انتقال داده را که از تقسیم مجموع سایز داده هایی که ارسال شده به کل زمان انتقال (زمان انتقال را باید به وسیله فراخوانی های سیستمی بدست بیاورید) حاصل میشود در ترمینال سرور نمایش دهید.

- **درخواست های پاک کردن و تغییر نام فایل های share شده:**

```
@remove Felan.txt
```

```
@rename Felan.txt newname
```

سرور با دریافت چنین درخواستی به شرطی که فرستنده درخواست صاحب فایل باشد فایل نام برده را از دایرکتوری خود پاک میکند یا نام آن را عوض میکند. (دقت کنید خطا های احتمالی باید به اطلاع کاربر برسند و در هر دو ترمینال سرور و کلاینت نتیجه درخواست به نمایش گذاشته شود. خطاهای احتمالی میتواند این باشد که مثلا درخواست دهنده صاحب فایل نباشد که در این صوریت اجازه تغییر فایل را نخواهد داشت یا اینکه فایل نام برده شده موجود نباشد یا اینکه فایلی با نام جدید ذکر شده قبلا موجود باشد)

- **درخواست فرستادن پیام به همکاران:**

```
@msg cli#i message_text
```

سرور در هر شرایطی (حتی اگر در حال دریافت و ارسال فایل باشد) به این درخواست پاسخ میدهد و client مقصد در پنجره خود پیام را به صورت زیر مشاهده میکند:

```
Cli#i:client_name said:
message_text
```

پس کلاینت ها در هر لحظه ای ممکن است از طرف یکی از همکارانشان پیامی دریافت کنند(به این مورد خوب دقت کنید). اما در حالتی که مقصد مورد نظر در حال انجام یک تراکنش سنگین باشد (در حال دریافت فایل از سرور یا ارسال فایل به سرور باشد) سرور پیام را به مقصد نمیرساند و اگر این حالت پیش بیاید به مبدا اطلاع داده میشود و کلاینت مبدا این خطا را در ترمینال خود نشان میدهد.

تمام پیام هایی که بین کلاینت ها رد و بدل میشود در پنجره سرور توسط رئیس شرکت به صورت زیر مشاهده میشود:

```
Cli#i:i_name to cli#j:j_name:
message_text
```

همچنین log (اطلاعات) کلیه درخواست هایی که از کلاینت ها به سرور میرسد نیز در ترمینال سرور قابل مشاهده است.

- **درخواست قطع ارتباط از سرور:**

```
@dc
```

با این دستور ارتباط کلاینت مورد نظر با سرور قطع میشود ، همچنین تمام فایل هایی که قبلا توسط این کلاینت share شده بود نیز از دایرکتوری سرور پاک میشود و کلاینت در شرایطی قرار میگیرد که مجددا میتواند دستور اتصال به همین سرور یا سرور دیگری را بدهد.

امکانات سرور:

رئیس شرکت نیز میتواند از طریق ترمینال سرور client ها را از شبکه اخراج کند:

```
@kick cli#i
```

سرور باید در هر شرایطی پاسخگو باشد.اگر در حال دریافت و ارسال فایل باشد به درخواست های از نوع دریافت و ارسال فایل جواب دهد که در حال حاضر نمی تواند درخواست آنها را انجام دهد ولی درخواست های دیگر

(درخواست های غیر ارسال و دریافت فایل) را در هر شرایطی انجام دهد. پیاده سازی این پاسخ گویی همزمان به کمک فراخوان سیستمی select بین file descriptor کلاینت های متصل به سرور انجام میشود (راجع به select در اینترنت تحقیق کنید):

سرویس هایی که سرور ارائه میدهد به ۲ دسته سرویس های سنگین و سبک تقسیم میشوند. سرویس های دریافت و ارسال فایل سرویس های سنگین محسوب شده و همه سرویس های دیگر سبک محسوب میشوند. سرویس های دریافت و ارسال فایل نیز خود از انتقال چندین packet داده (هر packet معادل سایز دلخواه مناسب و ثابت مثلا ۲۵۶ بایت و پروتکل های انتقال قبلا توضیح داده شده اند.) تشکیل میشوند که هر انتقال packet را میتوان یک سرویس سبک محسوب نمود.

همزمانی به این صورت انجام میشود که سرور در هر بار سرکشی select برای هر file descriptor حد اکثر یک کار سبک انجام میدهد.

مثال: سرور در حالی که در سرکشی های مداوم در حال ارسال packet های یک فایل به کلاینت ۰ است، کلاینت ۱ درخواست انتقال فایل میدهد و کلاینت ۲ درخواست ارسال پیام میکند در سرکشی بعدی به کلاینت ۱ خبر داده میشود که در حال حاضر لود سیستم بالاست پیام کلاینت ۲ دریافت شده به مقصد فرستاده میشود و packet بعدی از فایلی که تا به حال قسمتی از آن برای کلاینت ۰ فرستاده شده است به کلاینت ۰ ارسال میشود.

نکات پایانی:

- هر کلاینت از زمان فرستادن یک درخواست تا زمان دریافت کامل پاسخ آن نباید بتواند توسط ترمینال خود درخواست دیگری بفرستد. این مورد را میتوانید در طرف کلاینت handle کنید. (ولی ممکن است در این زمان پیامی از یکی از همکارانش دریافت کند . مگر در حال انجام کار سنگین با سرور باشد که در این صورت اگر یکی از همکاران بخواهد پیام به این کلاینت بفرستد سرور به فرستنده اطلاع میدهد که کلاینت مورد نظر در حال حاضر مشغول انتقال فایل میباشد)
- کد های شما که باید به زبان C نوشته شده باشد توسط کامپایلر gcc کامپایل شده و تحت سیستم عامل لینوکس اجرا میشوند.

- همه پروژه های درس در قالب گروه های دو نفره است و گروه ها تا پایان ترم عوض نمیشوند. لطفا در انتخاب همگروهی دقت کنید.
- هر دو نفر گروه باید تسلط لازم را به پروژه داشته باشند. افراد گروه لزوما نمره مساوی کسب نخواهند کرد.
- Log هایی که باید در ترمینال های کلاینت و سرور نمایش داده شود اغلب بیان شده است، در مواردی که بیان نشده است هر جا که تشخیص میدهید که نشان دادن اطلاعات در ترمینال باعث روشن شدن بیشتر روند کارتان میشود، log ها را نشان دهید. Log دادن هویت کار شماست. واضح است که برنامه نویسی ماژولار و با قاعده ، تمیز بودن کد، روشن تر کردن هرچه بیشتر روند کار بوسیله log دادن و روی هم رفته کیفیت کار شما در روند نمره دهی به شما تاثیر گذار است و تشخیص این امر کاملا بر عهده دستیار آموزشی است که پروژه را از شما تحویل میگیرد.
- در این پروژه مجاز به استفاده از تکنیک های Multithreading و Multiprocessing نیستید.
- برای درک بیشتر شما از صورت پروژه کلاسی برگزار خواهد شد که مفاهیم پروژه برای شما تشریح و مشکلات شما پاسخ داده میشود. همچنین در سایت درس انجمنی ساخته میشود که میتوانید سوالات خود را در آن مطرح کنید. در صورت عدم شرکت در کلاس یا بررسی نکردن انجمن، مسئولیت از دست دادن کلیه نکاتی که مطرح خواهند شد با شماست.
- تمام عملیات باید توسط فراخوانی های سیستمی مانند create، open، read، write، close و ... انجام شوند و استفاده از توابع کتابخانه ای (حتی کتابخانه استاندارد C) مانند :

  fopen – fclose – fprintf – fscanf – printf – scanf – perror

  مجاز نیست.(حتی اگر به دلیل debug یا ... این توابع را در کدی که برای ما آپلود میکنید قرار داده باشید نمره منفی دریافت خواهید کرد) ملاک این که یک تابع فراخوانی سیستمی است یا خیر این است که بتوانید نام این تابع را در لیست فراخوانی های سیستمی که در بخش دوم manpage لینوکس به آدرس زیر است پیدا کنید.

http://linux.die.net/man/2

- توابع کتابخانه ای که با فراخوانی های سیستمی قابل پیاده سازی نیستند مانند strcat، strcopy، atoi و ... مجاز هستند.
- برنامه شما باید نسبت به ورودی های مختلف عکس العمل مناسب داشته باشد. (Error Handling)

- فقط فایل های c. خود را در یک فایل zip با نامی به فرمت studentid1-studentid2.zip قرار داده در سایت درس آپلود کنید. (اگر این مورد را رعایت نکنید نمره منفی دریافت خواهید کرد)
- با هر گونه مشابهت در کد ها برخورد جدی خواهد شد.

موفق باشید.