آشنایی با فراخوان سیستمی[1]

موعد تحویل: ۲۴ مهر ۱۳۹۵

در این پروژه شما قرار است یک سیستم چت همتابه‌همتا[2] ی غیرامن را به‌کمک Socket Programming و فراخوان‌های سیستمی در زبان C پیاده‌سازی کنید. این سیستم می‌تواند با استفاده از یک سرور مرکزی به عنوان tracker کار کند اما در صورتی که برای سرور مشکلی پیش بیاید این قابلیت را نیز دارد که به‌شکل کاملاً غیرمتمرکز به فعالیت خود ادامه دهد.

در حالت غیرمتمرکز از آن‌جا که سروری وجود ندارد که تمام اطلاعات را در اختیار داشته باشد، تمام کلاینتها باید به‌محض ورود کاربر به آن‌ها، اطلاعات (شامل نام کاربر و پورتی که بر آن listen میکند) را از طریق Network Broadcast به اطلاع سایر کلاینت‌ها برسانند. (به پورت M که در ادامه خواهید دید.)

در صورتی که سرور مرکزی وجود داشته باشد، این سرور یک بار در ثانیه پیغام Heartbeat را در شبکه منتشر خواهد کرد (به پورت N) تا کلاینت‌ها از وجود و صحت کارکرد آن مطلع شوند (در این پیغام پورتی که سرور برای پاسخ دهی به درخواست‌های جستجو به آن listen میکند، آمده). به علاوه زمانی که کلاینت به شبکه دارای سرور مرکزی وارد میشود یا tracker پس از مدتی به شبکه اضافه میشود، کلاینت باید اطلاعات کاربر خود و پورتی که بر روی آن listen میکند را برای سرور با همان پورتی که در heartbeat آمده بفرستد تا سرور از وجود کاربر مطلع شود.

برای فرستادن فایل یا پیغام به سایر کاربران به‌صورت زیر عمل می‌شود:

- در صورت وجود سرور، فرستنده،IP ی کاربر گیرنده و پورتی را که کلاینت روی آن listenمی‌کند از سرور درخواست می‌کند و سپس پیغام خود را به کاربر گیرنده می‌فرستد.
- در صورت نبود سرور، فرستنده باید به‌طریقی دیگر از وجود گیرنده و آدرس وی مطلع شود و سپس پیغام خود را به کاربر گیرنده ارسال کند. تصمیم‌گیری در مورد نحوه‌ی پیاده‌سازی این مکانیزم با دانشجو است. (مثال: برای فرستادن یک فایل منطقی‌تر این است که تنها اگر از online بودن گیرنده اطمینان داریم، تمام فایل را روی شبکه بفرستیم زیرا در غیر این صورت منابع را هدر خواهیم داد. (راهنمایی: استفاده از پورت M برای پیدا کردن گیرنده))
- دقت کنید که پیغام‌های ارسال شده به کاربران offline باید در کلاینت فرستنده باقی بماند تا کاربر مربوطه online شود (که با broadcastوی مشخص خواهد شد) و پس از آن پیغام بازارسال شود.

---

[1] System Call
[2] Peer to Peer

نکته‌ی مهم دیگری که در کد کلاینت و سرور باید رعایت کنید این است که به‌کمک فراخوان سیستمی select، تمام I/O ها به شکل ناهمگام[3] انجام شوند و هیچ بخشی از کدتان blockingنباشد.

سرور باید این‌گونه اجرا شود:

```
./server --server-broadcasts-to N --clients-broadcast-to M
```

و اجرای آن تا terminate کردن آن ادامه پیدا می‌کند. توجه داشته باشید از آن‌جا که سرور به صورت single thread اجرا می‌شود، برای یک بار در ثانیه heartbeat دادن نیاز به استفاده از signal و alert خواهید داشت - البته اگر راه‌حلِ درست دیگری هم داشته باشید، کاملاً پذیرفته است.

اجرای کلاینت‌ها هم مانند سرور خواهد بود:

```
./client --server-broadcasts-to N --clients-broadcast-to M
```

پس از آن کاربر دستور login username را با username خود در ورودی استاندارد می‌نویسد و وارد سیستم می‌شود. در طول مدتی که کاربر online است هر پیامی که برای وی فرستاده شود باید به‌شکل مناسب و همراه با نام فرستنده نمایش داده شود. کاربر پیش از زدن دستور exit و بازگشت به مرحله‌ی login می‌تواند برای فرستادن پیغام به کاربری دیگر دستور زیر را وارد کند:

```
send receiver_username message
```

و به‌وسیله‌ی آن message را (که می‌تواند شامل کاراکتر فاصله باشد) به username (که فاصله ندارد) بفرستد. همچنین برای فرستادن فایل به کاربر دیگر از دستور زیر استفاده می‌شود:

```
send_file receiver_username file_address
```

که در نتیجه‌ی آن فایل برای گیرنده فرستاده خواهد شد (دقت داشته باشید اندازه‌ی فایل ارسالی می‌تواند بزرگ باشد و نباید دچار مشکل حافظه شوید).

دستور زیر نیز وضعیت و تعداد پیام‌های ارسال‌نشده را نشان می‌دهد: `stats`

خروجی نمونه‌ی این دستور به‌شکل زیر است:

```
2 message(s) and 1 file(s) for sepehr
1 message(s) and 0 file(s) for sameni
```

[3] Asynchronous

خلاصه:

سرور heartbeat را بر روی پورت N هر ثانیه می‌فرستد و بر روی پورت A - که در heartbeat اعلام می‌کند - منتظر درخواست‌های جستجو برای IP و پورت کاربر از طرف دیگر کاربران است

کلاینت‌ها با استفاده از پورت M اطلاعات خود را به شبکه (دیگر کلاینت ها) میدهند و بر روی پورت B هم منتظر پیام هستند (دقت کنید که برای هر کلاینت پورت B فرق کند که بتوان کل سیستم را بر روی یک دستگاه تست کرد)

سناریوهایی که باید برای سیستم در نظر بگیرید:

۱. زمانی که شبکه دارای tracker هست و یک client جدید وارد آن میشود.

۲. زمانی که شبکه بدون tracker هست و یک client جدید وارد آن میشود.

۳. زمانی که شبکه بدون tracker هست و tracker به آن اضافه میشود.

حالت‌های ارسال داده، هشت حالت هست، دو حالت برای online بودن یا نبودن گیرنده ضرب در دو حالت برای نوع پیام (فایل یا متن بودن پیام) ضرب در دو حالت هم برای وضعیت شبکه یعنی حاوی tracker بودن یا نبودن که البته با پیاده‌سازی خوب علنا شامل دو حالت کلی است.

نکات پایانی:

- کدهایتان باید به زبان C نوشته شوند و با gcc قابل کامپایل باشند.
- ممکن است پروژه‌های بعدی در ادامه‌ی این پروژه باشند. پس حتماً این پروژه را کامل انجام دهید.
- حتما در یکی از دو جلسه‌ی توجیهی حضور داشته باشید. نکاتی که در کلاس یا فروم درس مطرح می‌شوند جزو پروژه هستند.
- این پروژه انفرادی‌ست.
- Log های مورد نیاز را چاپ کنید. (قطع و وصل کلاینت‌ها، درخواست‌ها، انتقال فایل‌ها و ...). این نکته در زمان تحویل اهمیت دارد و بخشی از نمره‌ی شما را تشکیل می‌دهد.
- پیاده‌سازی باید توسط فراخوانی‌های سیستمی مانند write،read،open،create و ... انجام شود و استفاده از توابع کتاب‌خانه‌ای (حتی کتاب‌خانه‌ی استاندارد) مانند fopen،fprintf و ... مجاز نیست. حتی اگر به هر دلیلی مانند debug این توابع را در کدی که برای ما آپلود می‌کنید قرار داده باشید نمره‌ی منفی دریافت خواهید کرد (ملاک این‌که یک تابع فراخوانی سیستمی است یا خیر این است که بتوانید نام این تابع را در لیست فراخوانی‌های سیستمی در بخش دوم لینوکس به آدرس http://linux.die.net/man پیدا کنید).

- توابع کتاب‌خانه‌ای که با فراخوانی‌های سیستمی قابل پیاده‌سازی نیستند مانند strcat،atoi و ... مجاز هستند.
- کد کسی را کپی نکنید. حتی یک تابع.
- تمیز کد بزنید.
- آموخته‌های خود را در درس برنامه‌نویسی پیشرفته فراموش نکنید، makefile یا cmake داشته باشید (نداشتن آن نمره‌ی منفی دارد)
- این حقیقت که در حال حاضر low level کد می‌زنید به هیچ وجه توجیه‌کننده استفاده‌ی نادرست از متغیرهای global نیست تا حد امکان از آن‌ها بپرهیزید و تنها با دلایل معقول از آن‌ها استفاده کنید.
- Memory یا هر منبع دیگری را نشت ندهید، این موضوع در زمان تحویل بررسی خواهد شد.
- تنها تابع‌هایی که از system call استفاده می‌کنند و نیازی به پیاده‌سازی آن‌ها نیست free و malloc (و هم خانواده‌های آن مثل realloc) هستند، البته پیاده‌سازی آن‌ها نمره‌ی امتیازی خواهد داشت.
- برای آشنایی با Socket Programming می‌توانید به صفحات زیر مراجعه کنید:

  http://beej.us/guide/bgnet/output/html/multipage/clientserver.html

  http://beej.us/guide/bgnet/output/html/multipage/advanced.html#broadcast