

پاسخ سوالات مینی پروژه سیستم عامل

محمد مهدی ناصری

۹۴۲۳۱۱۵

پاسخ سوال یک

بخش یک

اطلاعات مربوط به PCB هر فرآیند در proc.h آمده است که به شرح زیر است:

```
// Per-process state
struct proc {
    uint sz;           // Size of process memory (bytes)
    pde_t* pgdir;      // Page table
    char *kstack;      // Bottom of kernel stack for this process
    enum procstate state; // Process state
    int pid;           // Process ID
    struct proc *parent; // Parent process
    struct trapframe *tf; // Trap frame for current syscall
    struct context *context; // switch() here to run process
    void *chan;        // If non-zero, sleeping on chan
    int killed;        // If non-zero, have been killed
    struct file *ofile[NOFILE]; // Open files
    struct inode *cwd; // Current directory
    char name[16];     // Process name (debugging)
};
```

توضیحاتی در ارتباط با برخی از متغیرها در زیر آمده است:

pgdir: جدول صفحه‌ی فرآیند در واقع ساختمان داده‌ای است که برای map کردن آدرس‌های مجازی به فیزیکی به کار می‌رود.

kstack: انتهای stack کرنل برای این فرآیند؛ هر فرآیند دارای دو stack شامل kernel stack و user stack می‌باشد که در هنگام اجرا شدن در فضای user، kernel stack خالی می‌باشد.
tf: در هنگام وارد شدن به کرنل وضعیت user space را ذخیره می‌کند.

بخش دو

sz: سائز حافظه‌ی هر فرآیند را نگهداری می‌کند تا با دانستن شروع حافظه بتوان انتهای آن را محاسبه کرد.
state: وضعیت فعلی هر فرآیند را مشخص می‌کند.

context: وضعیت کرنل را در زمان context switch ذخیره می‌کند.

ofile: فایل‌هایی که توسط این فرآیند باز هستند را مشخص می‌کند.

killed: اگر مقداری غیر از صفر داشته باشد به معنای آن است که به عنوان مثال توسط فرآیندی دیگر kill شده است.

پاسخ سوال سه

به طور کلی در هر بار run گرفتن مقدار متفاوتی برای زمان اندازه گیری اعلام می شد که با هم اختلاف زیادی داشتند و به طور دقیق قابل استناد نبود اما مورد انتظار آن است که برای مقادیر خیلی کم M مقدار runtime افزایش یابد زیرا تعداد fork ها خیلی زیاد شده و عملاً دیگر صرفی ندارد تا هر تکه کوچک از آرایه را نیز به صورت موازی sort کرد چرا که مجموع زمان به وجود آمدن و هندل کردن خود fork ها از مجموع زمانی که به صورت sequential اجرا صورت می گرفت بیشتر شده است.

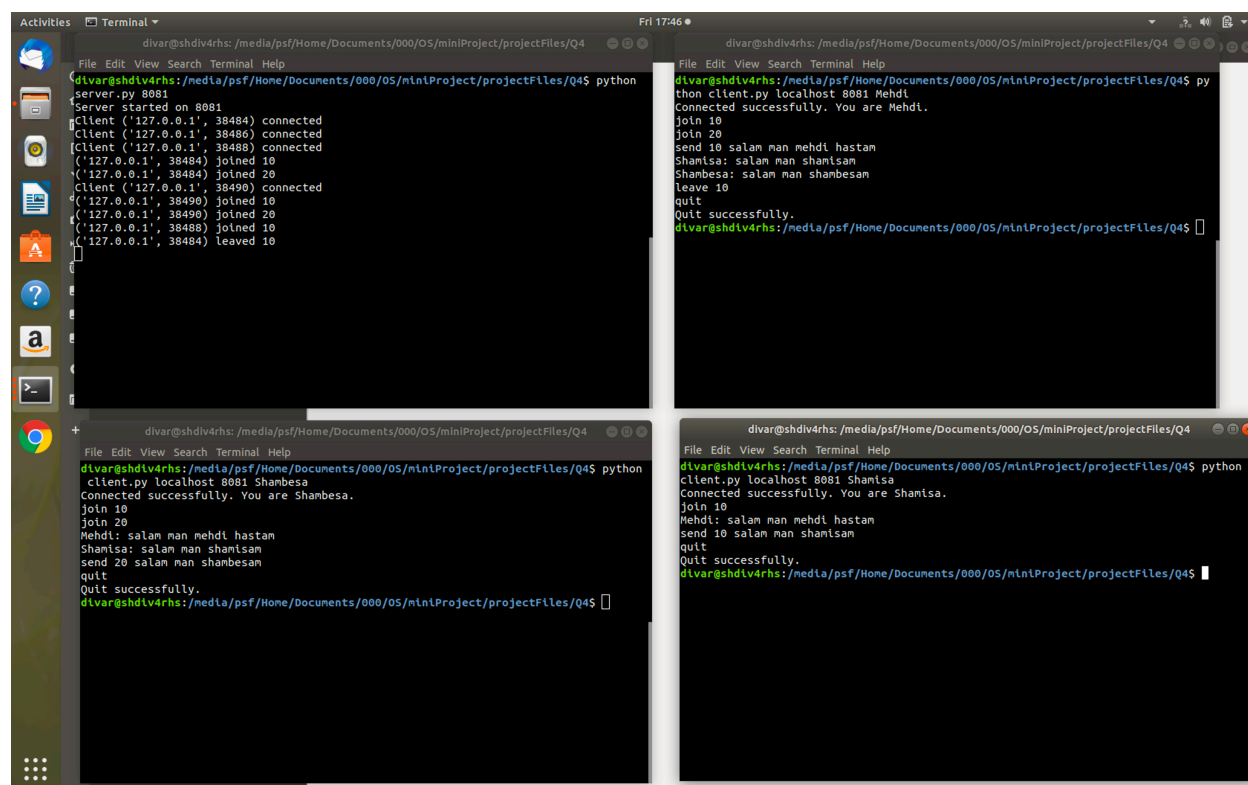
تست کردن سوال چهار

برای تست کردن چت سرور از ۳ کلاینت و ۱ سرور استفاده شده است. برای ران کردن کد سرور و کلاینت باید به این صورت عمل کرد:

```
python server.py portNumber
```

```
python client.py serverIP serverPort clientName
```

به عنوان مثال در تست بالا سه کلاینت با نام های Mehdi و Shamisa و Shambesa وجود دارند. هر سه آن ها عضو گروه ۱۰ می شوند ولی فقط Mehdi و Shambesa عضو گروه ۲۰ می شوند. مشاهده می شود هنگامی که از طریق گروه ۱۰ پیامی ارسال می شود برای دو عضو دیگر نمایش داده می شود ولی در گروه ۲۰ از آن جایی که تنها دو عضو دارد هنگامی که پیامی از طریق یکی از اعضا ارسال شود صرفاً برای عضو دیگر گروه ارسال می شود.



The image displays three terminal windows from a Linux desktop environment, showing the execution of a chat server and its clients.

Top Left Terminal: Shows the server running `python server.py 8081`. It logs connections from clients at `127.0.0.1` on ports 38484, 38486, 38488, 38490, 38498, and 38484. The clients join groups 10 and 20, and the server logs their messages.

Top Right Terminal: Shows a client running `python client.py localhost 8081 Mehdi`. It connects successfully and sends the message "salam man mehdi hastan".

Bottom Left Terminal: Shows a client running `python client.py localhost 8081 Shambesa`. It connects successfully and sends the message "salam man shambesan".

Bottom Right Terminal: Shows a client running `python client.py localhost 8081 Shamisa`. It connects successfully and sends the message "salam man shamsam".