

آریا خلیق
۹۵۲۴۰۱۴
bartararya@gmail.com
مینی پروژه اول

نکته: فایل git تمرین دوم جداگانه و در فایل زیپ git.zip وجود دارد که در صورت تمایل آن را extract کنید تا log commit های تمرین دوم را مشاهده نمایید. مابقی تغییرات داخل فایل git اصلی آورده شده.

سوال اول:

(۱)

در این سیستم عامل PCB داخل فایل proc.h تعریف شده. PCB به صورت یک struct در C می باشد.

```
// 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)
};
```

(۲)

:SZ

unit یک تایپ از نوع unsigned int می باشد که به صورت زیر در فایل types.h تعریف شده:

```
typedef unsigned int uint;
```

خود SZ هم نشان دهنده میزان حجم یک فرایند (بر اساس بایت) در حافظه می باشد.

:state

تایپ متغیر state به صورت enum می باشد. تعریف procstate به صورت زیر است:

```
enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };
```

پس یک فرایند می تواند state ای از موارد بالا داشته باشد.

:Context

خود متغیر context یک نوع struct می باشد که زمانی که CPU از یک فرایند پس گرفته می شود وضعیت فعلی فرایند در آن ذخیره می شود که دوباره وقتی CPU را به دست آورد لود شده و از حالت قبلی ادامه دهد.

```
struct context { uint edi; uint esi; uint ebx; uint ebp; uint eip; };
```

:ofile

این متغیر یک آرایه به طول NOFILE می‌باشد که با مقدار زیر تعریف شده است:

```
#define NOFILE 16      // open files per process
```

struct فایل به صورت زیر می‌باشد:

```
struct file {  
    enum { FD_NONE, FD_PIPE, FD_INODE } type;  
    int ref; // reference count  
    char readable;  
    char writable;  
    struct pipe *pipe;  
    struct inode *ip;  
    uint off;  
};
```

با توجه به دو قطعه کد زیر:

```
for(i = 0; i < NOFILE; i++)  
    if(proc->ofile[i])  
        np->ofile[i] = filedup(proc->ofile[i]);  
np->cwd = idup(proc->cwd);
```

9

```
for(fd = 0; fd < NOFILE; fd++){  
    if(proc->ofile[fd]){  
        fileclose(proc->ofile[fd]);  
        proc->ofile[fd] = 0;  
    }  
}
```

proc از struct درون ofile استفاده می‌کند.

:Killed

همان طور که در کامنت توضیح داده شده است اگر هر مقدار غیر 0 داشته باشد به معنی کشته شدن و اتمام فرایند است.

```
int killed;          // If non-zero, have been killed
```

سوال دوم:

در این سوال یک بده بستان بین استفاده از پردازش موازی و همزمان و overhead وجود دارد که با کم شدن M از یک مقدار خاص و نزدیک شدن آن به ∞ این overhead افزایش یافته و باعث افزایش زمان اجرای برنامه می‌شود.