آریا خلیق
۹۵۲۴۰۱۴
bartararya@gmail.com
مینی‌پروژه اول

سوال اول:

۱)

در این سیستم‌عامل PCB داخل فایل proc.h تعریف شده. PCB به صورت یک struct در C می‌باشد.

```
// 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;     // swtch() 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 می‌باشد که به صورت زیر در فایل types.h تعریف شده:

typedef unsigned int uint;

خود sz هم نشان‌دهنده میزان حجم یک فرایند (بر اساس بایت) در حافظه می‌باشد.

state:

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

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

پس یک فرایند می‌تواند stateای از موارد بالا داشته باشد.

Context:

خود متغیر context یک نوع struct می‌باشد که زمانی که CPU از یک فرایند پس گرفته می‌شود وضعیت فعلی فرایند در آن ذخیره می‌شود 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);
```



```
  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:

همان‌طور 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 ۰ این overhead افزایش یافته و باعث افزایش زمان اجرای برنامه می‌شود.