به نام خدا

فاز ۱ پروژه سیستم عامل

محمد چوپان ۹۸۳۱۱۲۵

- **int getProcCount(void)**

در این فراخوانی سیستمی از شما می‌خواهیم که تعداد ***پردازه‌هایی (processes) را که در سیستم*** در لحظه فراخوانی این فراخوانی سیستمی وجود دارند را برگردانید.

همچنین پس از ساخت این فراخوانی سیستمی از شما می خواهیم که یک فایل به نام getProcCountTest.c ایجاد کنید که بتوانید صحت کارکرد این فراخوانی سیستمی را چک کنید. دقت کنید که اگر قبل از اجرای برنامه getProcCountTest در سیستم xv6 شما مثلا ۲۰ پردازه وجود داشته باشد، getProcCount عدد ۲۱ را برمی‌گرداند چرا که خود برنامه getProcCountTest یک پردازه جدید است.

ابتدا در فایل های زیر تابع مورد نظر را تعریف می کنیم.

Syscall.c,defs.h,usys.h,syscall.h,user.h,

سپس در فایل sysproc.c تابع خود را تعریف میکنیم.

```
sys_ProcCount(void){
  return ProcCount();
}
```

و در نهایت در فایل proc.c قسمت اصلی تابع را پیاده سازی میکنیم.

```
ProcCount(void){
  struct proc *p;

  int counter=0;
  p=ptable.proc;
  for(int i=0; i < NPROC; i++){
    if(p[i].state !=UNUSED){
      counter++;
      cprintf("%s\n",p[i].name);

    }
  }


  return counter;
}
```

در این تابع ابتدا شمارنده خود را برای شمارش تعداد پردازه ها تعریف میکنیم که در ابتدا صفر است.

سپس یک struct تعریف کرده و آن را برابر آرایه کل پردازه ها قرار می دهیم.

در حلقه ای در طول این ساختار حرکت کرده و پردازه هایی که در حال اجرا و یا zombie و هر حالتی جز غیر استفاده دارند را

شمرده و نام آن ها را چاپ میکنیم.

در نهایت نیز آن تعداد پردازه ها را چاپ میکنیم.

برای تست کردن این systemcall نیز یک فایل تست یه صورت زیر تعریف کرده و در make هم تغیراتی ایجاد میکنیم.

```
#include "types.h"
#include "stat.h"
#include "user.h"
int main (void){
    printf(1,"number of process
    is : %d \n",ProcCount());
    printf(1,"\nsuccess\n");
    exit();
}
```

- **`int getReadCount(void)`**

در این فراخوانی سیستمی از شما می‌خواهیم که تعداد دفعاتی که فراخوانی سیستمی ***Read*** توسط هر پردازه‌ی دیگر کاربر فراخوانی شده است (از زمانی که کرنل بوت شده) را برگردانید.

همچنین پس از ساخت این فراخوانی سیستمی از شما می خواهیم که یک فایل به نام `getReadCountTest.c` ایجاد کنید که بتوانید صحت کارکرد این فراخوانی سیستمی را چک کنید.

در این قسمت تعریف ها را همانند بالا در فایل های مشابه ایجاد میکنیم.

سپس در فایل sysfile.c متغیری برای شمارش تعداد Readها تعریف کرده و مقدار آن را در تابع sys_read افزایش می دهیم.سپس در تابع اصلی آن را چاپ میکنیم.

```
int
sys_read(void)
{
  ReadCount++;
  struct file *f;
  int n;
  char *p;

  if(argfd(0, 0, &f) < 0 || argint(2, &n) < 0 || arg
    return -1;
  return fileread(f, p, n);
}
```

```
#include "fcntl.h"
int ReadCount=0;
// Fetch the nth wo
```

```
int sys_getReadCount(void){
 return getReadCount();
}
```

فایل sysproc.c

صرفا تعریف تابع:

```
int getReadCount(void){

  return ReadCount;
}
```

```
int sys_getReadCount(void);
```

و در نهایت همانند بالا فایل تست:

```
#include "types.h"
#include "stat.h"
#include "user.h"
int main (void){
    printf(1,"number of Read is
    : %d \n",getReadCount());

    printf(1,"\nsuccess\n");
    exit();
}
```

برای هر دو makeFile را نیز تغییر می دهیم.

```
UPROGS=\
    _cat\
    _echo\
    _forktest\
    _grep\
    _init\
    _kill\
    _ln\
    _ls\
    _mkdir\
    _rm\
    _sh\
    _stressfs\
    _usertests\
    _wc\
    _zombie\
    _getProcCountTest\
    _getReadCountTest\
```

```
EXTRA=\
    mkfs.c ulib.c user.h cat.c
    ln.c ls.c mkdir.c rm.c stre
    printf.c umalloc.c\
    README dot-bochsrc *.pl toc
    .gdbinit.tmpl gdbutil\
    getProcCountTest.c\
    getReadCountTest.c\
```