

” به نام خداوند جان و خرد ”

فاز اول پروژه سیستم عامل  
گزارشکار

نام دانشجو: کیانا آقاگیری 9831006

نام استاد : دکتر جوادی

تاریخ تحویل پروژه : 1400/07/30

ابتدا یک تابع `getProcCount` در `proc.c` می‌سازیم. در این تابع، روی کل `process`‌ها حرکت میکند و به ازای آنهایی که `UNUESD` نباشند شمارنده را یکی زیاد میکند و در نهایت آن را چاپ کرده و `return` میکند.

```
int getProcCount(void){
    struct proc *p;
    int counter=0;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
        if(p->state != UNUSED) counter++;
    }
    cprintf("%d",counter);

    return counter;
}
```

سپس در فایل‌های دیگر آن را اضافه می‌کنیم تا این `system call` را بشناسند. فایل‌های دیگر:

- 1.MakeFile
- 2.defs.h
- 3.syscall.h
- 4.sysfile.c
- 5.sysproc.c
- 6.user.h
- 7.usys.S
- 8.syscall.c

حال برای اضافه کردن `system call` دوم تابع `getReadCount` را در `sysproc.c` تعریف می‌کنیم و یک متغیر `extern` می‌سازیم که این تابع آن را چاپ و `return` میکند.

```
extern int readCount;
int sys_getReadCount(void)
{
    cprintf("%d",readCount);
    return readCount;
}
```

حال در فایل `sysfile.c` تابعی به نام `sys_read` وجود دارد. متغیر `readCount` را در بیرون این تابع هم تعریف می‌کنیم و در این تابع می‌زنیم یکی اضافه شود. بدین صورت تعداد دفعاتی که فراخوانی سیستمی **Read** توسط هر پردازش دیگر کاربر فراخوانی شده است (از زمانی که کرنل بوت شده) را برمیگرداند.

```

int readCount = 0;
int
sys_read(void)
{
    readCount++;
    struct file *f;
    int n;
    char *p;

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

```

و دوباره در فایل های دیگر آن را اضافه میکنیم تا این system call را بشناسند.

سپس یک فایل تست برای آنها مینویسیم و درستی این system call را بررسی میکنیم.

getProcCountTest.c:

```

C getProcCountTest.c
1  #include "types.h"
2  #include "stat.h"
3  #include "user.h"
4
5  int main (void) {
6      getProcCount();
7      printf(1,"proc count success");
8      exit();
9  }

```

getReadCountTest.c:

```

C getReadCountTest.c
1  #include "types.h"
2  #include "stat.h"
3  #include "user.h"
4
5  int main (void) {
6      getReadCount();
7      printf(1,"read count success");
8      exit();
9  }

```

بدین صورت میبینیم پاسخ صحیح است:

```
kiana@ubuntu: ~/Desktop/vx6-public
QEMU
Machine View
README      2 2 2286
cat          2 3 16328
echo         2 4 15180
forktest    2 5 9484
grep         2 6 18548
init         2 7 15768
kill         2 8 15208
ln           2 9 15068
ls           2 10 17696
mkdir        2 11 15308
rm           2 12 15288
sh           2 13 27924
stressfs     2 14 16200
usertests    2 15 67308
wc           2 16 17060
zombie       2 17 14880
getProcCountTe 2 18 14920
getReadCountTe 2 19 14920
console      3 20 0
$ getProcCountTest
3proc count success$ getReadCountTest
70read count success$ getReadCountTest
87read count success$ getReadCountTest
104read count success$
```