” به نام خداوند جان و خرد”

فاز دوم پروژه سیستم عامل
گزارشکار

نام دانشجو: کیانا آقاکثیری 9831006

نام استاد : دکتر جوادی

تاریخ تحویل پروژه : 1400/10/03

در این فاز باید دو سیستم کال thread_wait و thread_create پیاده سازی کنیم. در proc.c دو تابع به همین نام ها تعریف میکنیم.

```
642  int
643  thread_create (void *stack)
644  {
645    int pid;
646    struct proc *curproc = myproc();
647    struct proc *np;
648    if ( (np = allocproc()) == 0)
649      return -1;
650
651      curproc->threads++;
652      np->stackTop = (int)((char*)stack + PGSIZE);
653    acquire(&ptable.lock);
654    np->pgdir = curproc->pgdir;
655    np->sz = curproc->sz;
656    release (&ptable.lock);
657
658    int bytesOnStack = curproc->stackTop - curproc->tf->esp;
659    np->tf->esp = np->stackTop - bytesOnStack;
660    memmove((void*)np->tf->esp, (void*) curproc->tf->esp, bytesOnStack);
661
662    np->parent = curproc;
663    *np->tf = *curproc->tf;
664    np->tf->eax = 0;
665    np->tf->esp = np->stackTop - bytesOnStack;
666    np->tf->ebp = np->stackTop - (curproc->stackTop - curproc->tf->ebp);
667
668    int i;
669    for(i = 0; i < NOFILE; i++)
670      if(curproc->ofile[i])
671        np->ofile[i] = filedup(curproc->ofile[i]);
672    np->cwd = idup(curproc->cwd);
673    safestrcpy(np->name, curproc->name, sizeof(curproc->name));
674    pid = np->pid;
675    acquire(&ptable.lock);
676    np->state = RUNNABLE;
677    release (&ptable.lock);
678
679    return pid;
```

```
682  int
683  thread_wait(void)
684  {
685    struct proc *p;
686    int havekids, pid;
687    struct proc *curproc = myproc();
688
689    acquire(&ptable. lock);
690    for(;;){
691      havekids = 0;
692      for (p = ptable.proc; p < &ptable.proc[NPROC]; p++){
693        if(p->parent != curproc)
694          continue;
695        if (p->threads != -1) // remember join only waits for child threads not child processes
696          continue;
697        havekids = 1;
698        if (p->state == ZOMBIE){
699          pid = p->pid;
700          kfree(p->kstack);
701          p->kstack = 0;
702
703          if (check_pgdir_share(p))
704            freevm(p->pgdir);
705
706          p->pid = 0;
707          p->parent = 0;
708          p->name[0] = 0;
709          p->killed = 0;
710          p->state = UNUSED;
711          p->stackTop = 0;
712          p->pgdir = 0;
713          p->threads = -1;
714
715          release(&ptable.lock);
716          return pid;
717        }
718      }
719      if (!havekids || curproc->killed){
720        release(&ptable.lock);
721        return -1;
722      }
723      sleep(curproc, &ptable.lock);
724    }
725  }
```

همانند فاز اول باید در فایل های دیگر آن را اضافه میکنیم تا این system call را بشناسند. فایل های دیگر :

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

همچنین چون page directory فرزند و پدر در thread برخلاف process یکی میباشد ،تابع های دیگری را در proc.c مانند growwproc و allocproc و exit را تغییر میدهیم. (در تابع exit وقتی که فرزند این تابع را صدا می زند یکی از تعداد thread های با فضای آدرس مشترک با این پدر ، کم می کنیم .)

(در تابع growproc باید sz که نشان دهنده size فضای آدرس است را برای همه threadها، update کنیم.)

درexec.c ، thread و stackTop را مقداردهی اولیه میکنیم وthread و stackTop را درstruct proc اضافه میکنیم.
در بالای proc.c هم lock تعریف میکنیم به نام thread.

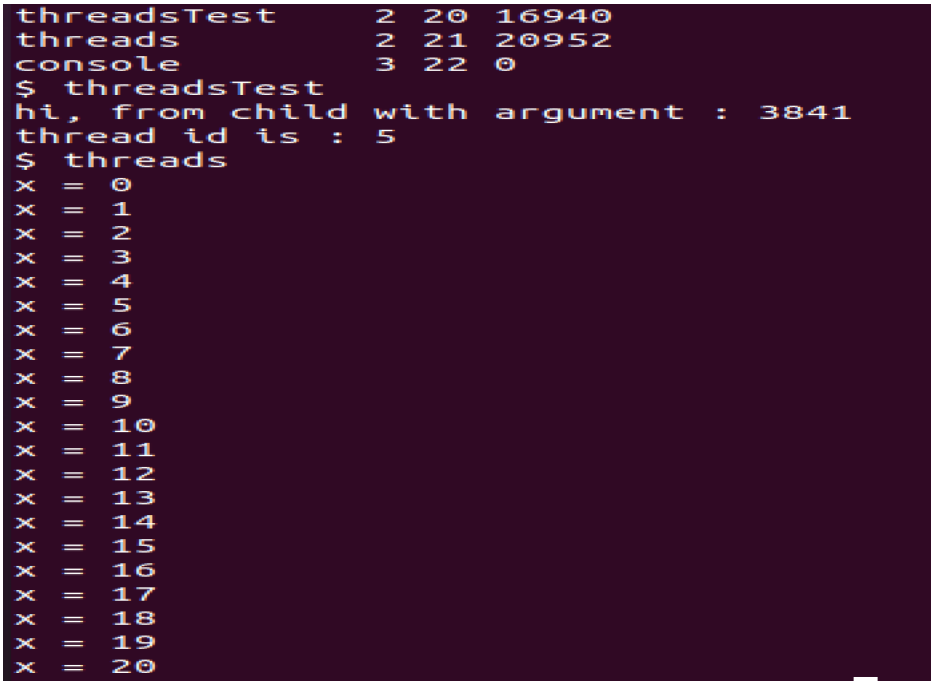
یک تابع برای چک کردن یکی بودن فضای آدرس پدر و فرزند با نام check_pgdir_share نیز داریم اگر غیر صفر برگرداند که page directoryرو خالی میکنیم اما در غیر این صورت نباید اینکار را بکنیم چون یک thread با فضای آدرس یکسان با این پدر وجود دارد که هنوز کارش تمام نشده است .

۲فایل تست threads و threadsTest میسازیم که در threads یک متغیر x را افزایش می دهیم که از thread_create استفاده می کنیم در این حالت باتوجه به اینکه فضای آدرس پدر و فرزند یکی میباشد پس تغییرات فرزند روی متغیر x برای والد هم قابل مشاهده است و مقدار x یکی یکی اضافه میشود . (sleep)

تست threadsTest هم برای تابع thread_creator داریم .در این تست مقدار این آرگومان را 3841است و در خروجی چاپ می شود .

بدین صورت پاسخ را مشاهده میکنیم:


hi, from child with argument : 3841
thread id is : 5
$ threads
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
x = 10
x = 11
x = 12
x = 13
x = 14
x = 15
x = 16
x = 17
x = 18


threadsTest 2 20 16940
threads 2 21 20952
console 3 22 0
$ threadsTest
hi, from child with argument : 3841
thread id is : 5
$ threads
x = 0
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
x = 10
x = 11
x = 12
x = 13
x = 14
x = 15
x = 16
x = 17
x = 18
x = 19
x = 20