پرهام هدایتی 400522301

محمدعلی خسروابادی 400521297

این کد یک نسخه ساده شده از فراخوانی سیستم کلون را در هسته سیستم عامل پیاده سازی می کند. هدف از این فراخوانی سیستم ایجاد یک thread جدید در همان فرآیند است. در اینجا توضیحی در مورد کد آمده است:

سیستم کال ها:

Clone:

```
void(*fcn)(void*, void*)
```

تابعی که به ترد میدهیم تا ان را اجرا کند که دو تا ارگومان دارد

یک پوینتر به تابع نشان دهنده نقطه شروع رشته جدید

```
void *arg1, void *arg2
```

ارگومان هایی که به تابع پاس میدهیم.

```
void* stack
```

پوینتر به ادرس شروع استکی که رشته ازش استفاده میکند

بدنه:

درواقع داریم با کلون برای پروسس p رشته جدید میسازیم پس فیلد numthread پروسس p رو یکی اضافه میکنیم رشته جدیدی که ساختیم رو اسمش رو thread میزاریم چون هر رشته فضای ادرسش با پروسس پرنتش مشترکه پس همه چیز های پرنت رو کپی میکنیم برای thread هم میزاریم فقط فلگ isthread رو 1 میکنیم(در فورک 0 میکنیم) بعدش هم استک رو مقداردهی اولیه میکنیم و در استک دو تا ارگومان های تابع و ادرس ریترن رو دخیره میکنیم(البته یه ادرس فیک میزاریم) بعد جدول رو قفل میکنیم تا از ریس کاندیشن جلوگیری کنیم استیت ترد یا رشته رو به RUNNABLE تغییر میدهیم و قفل را باز میکنیم

Join:

در تابع جوین هر پروسس به دنبال پروسه های فرزندی میگردد که به حالت زامبی درامده اند تا انهارا از جدول پاک کند

پس اول باید یه فور رو همه پروسس ها میزنیم که ترد های فرزندشو پیدا کنیم cp یه پوینتر به پروسس در حال اجرامونه و p یه پوینتر به پروسسی که قراره به عنوان ترد شناخته بشه و بقیه کد واضح هستش

توابع لایبرری :

این کد ها چون خیلی ساده هستند به جای توضیح متن خود کد را اورده ایم

```
int thread_create(void (*start_routine)(void *, void *), void* arg1, void*
arg2)
{
  void* stack;
  stack = malloc(PGSIZE);

  return clone(start_routine, arg1, arg2, stack);
}

int thread_join()
{
  void * stackPtr;
  int x = join(&stackPtr);
  return x;
}

int lock_init(lock_t *lk)
{
  lk->flag = 0;
  return 0;
}

void lock_acquire(lock_t *lk){
  while(xchg(&lk->flag, 1) != 0);
}

void lock_release(lock_t *lk){
  xchg(&lk->flag, 0);
}
```

تغییرات لازم برای اضافه کردن:

برای تعریف دو سیستم کال جدید به نام های clone و join ابتدا باید در فایل syscall.h انهارا اضافه کنیم و به انها یک شماره بدهیم

در فایل syscall.c انهارا تعریف میکنیم

برای دسترسی یوزر به این توابع باید توی usys.s SYSCALL(join) و SYSCALL(clone) را اضافه کنیم و همچنین در user.h به همراه ورودی هایشان نیز آن را تعریف کنیم.

و باید توابعی که به عنوان لایبرری تعریف کردیم مثل thread_create و ... باید در user.h تعریف شوند

برای استفاده از مفهوم lock واستفاده آن در فایل user.h یک استراکت به نام lock_t میسازیم که فقط با یک فلگ عمل میکند.

در فایل sysproc.c با استفاده از argint ورودی های پاس داده شده به تابع ها را دریافت میکنیم

تعریف تابع های جدید که در لایبرری باید در نظر گرفتیم باید در ulib.c قرار بگیرند

https://github.com/MAforgood/xv6-OS-Project