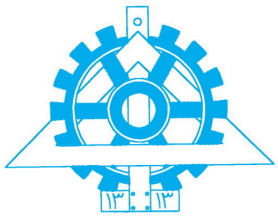


پروژه آزمایشگاه شماره ۳

دانشکده مهندسی برق و کامپیوتر

سیستم عامل - پاییز ۱۳۹۹

استاد : **دکتر کارگهی**

گروه ۱۷

اعضای گروه: دانشور امراللهی، علیرضا

توکلی، امین ستایش

---

۱. چرا فراخوانی sched منجر به فراخوانی scheduler می‌شود؟ (منظور، توضیح شیوه اجرای فرایند است)

وقتی که حالت یک پردازه به RUNNABLE تغییر پیدا می‌کند، تابع sched صدا زده می‌شود.

همانطور که در زیر در کد تابع می‌بینیم. از تابع swtch استفاده شده است که عمل کانتکست سوییچ را انجام می‌دهد.

یعنی کانتکست فعلی ذخیره می‌شود و سپس به scheduler سوییچ می‌کنیم.

یعنی scheduler جایگزین پردازه فعلی می‌شود و آن اجرا می‌شود تا پردازه را انتخاب کند و از حالت RUNNABLE به حالت RUNNING تبدیل کند.

```
void
sched(void)
{
 int intena;
 struct proc *p = myproc();

 if(!holding(&ptable.lock))
   panic("sched ptable.lock");
 if(mycpu()->ncli != 1)
   panic("sched locks");
```

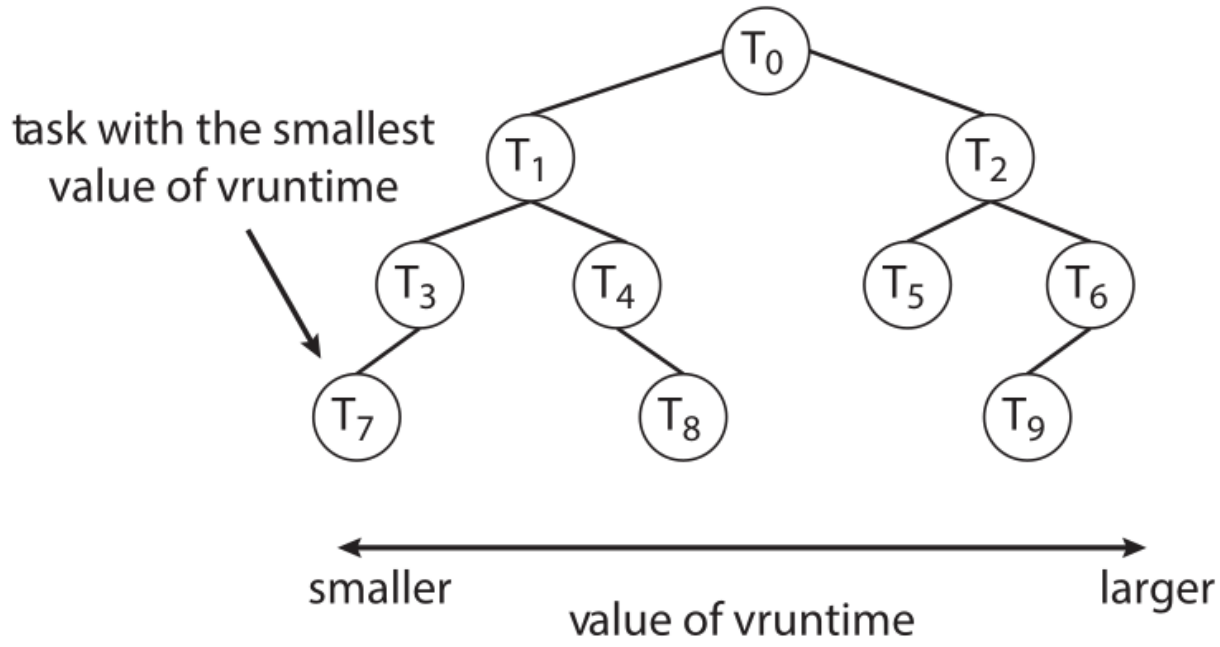
```
 if(p->state == RUNNING)
   panic("sched running");
 if(readeflags()&FL_IF)
   panic("sched interruptible");
 intena = mycpu()->intena;
 swtch(&p->context, mycpu()->scheduler);
 mycpu()->intena = intena;
}
```

۲. صف پردازه‌هایی که تنها منبعی که برای اجرا کم دارند پردازنده است، صف آماده یا صف اجرا نام دارد. در xv6 صف آماده مجزا وجود ندارد و از صف پردازه بدین منظور استفاده می‌شود. در زمان‌بند کاملا منصف در لینوکس، صف اجرا چه ساختاری دارد؟

همانطور که در کتاب درس هم به این موضوع اشاره شده است، در لینوکس از یک red black tree برای ساختار این صف استفاده می‌شود.

به این صورت که از vruntime به عنوان کلید استفاده شده است. (در لینوکس از vruntime برای اولویت‌بندی استفاده می‌شود که از ترکیب وزن و زمان اجرا به دست می‌اید.

شکل آن در زیر نمایش داده شده است:



۳. همانطور که در پروژه مشاهده شد، هر هسته پردازنده در xv6 یک زمان‌بند دارد. در لینوکس نیز به همین گونه است. این دو سیستم عامل را از منظر مشترک بودن یا مجزا بودن صف‌های زمان‌بندی بررسی نمایید.

یک مزیت و یک نقص صف مشترک نسبت به صف مجزا بیان کنید.

در xv6 فقط یک صف مشترک برای همه داریم که طبق حالت زیر تعریف شده است.

```
struct {
 struct spinlock lock;
 struct proc proc[NPROC];
} ptable;
```

در این struct یک صف از پردازه‌ها و یک قفل برای هندل کردن دسترسی‌های همزمان وجود دارد.

به این صورت که در هنگام استفاده قفل را اعمال می‌کنیم و سپس ptable را رها می‌کنیم.

اما در لینوکس هر پردازنده یک صف مخصوص به خود دارد.

همانطور که در کتاب اشاره شده است مزیت صف مشترک این است که نیاز به هندل کردن load پردازنده‌ها نداریم، یعنی مثلا نیاز به load balancing نداریم چون همه در یک صف هستند.

نقص صف مشترک این است که نیاز به بررسی دسترسی‌های همزمان به صف را داریم که از روش‌های locking برای این موضوع استفاده می‌شود.

۴. در هر اجرای حلقه برای مدتی وقفه فعال می‌شود. علت چیست؟ آیا در سیستم‌های تک‌هسته‌ای به آن نیاز است؟

حالتی را در نظر بگیریم که همه پردازه‌ها در حال گرفتن ورودی یا منتظر خروجی دادن هستند و یعنی هیچ پردازه RUNNABLE نداریم.

در این حالت اگر وقفه نداشته باشیم و غیرفعال باشد، هیچوقت ورودی و خروجی به پایان نمی‌رسد و برای اینکه به درستی انجام شود در هر حلقه وقفه برای مدتی فعال می‌شود تا این حالت پیش نیاید.

در سیستم‌های تک‌هسته‌ای هم نیاز داریم چون باز همین سناریو ممکن است پیش بیاید و باید وقفه داشته باشیم. یعنی این موضوع وابسته به تعداد هسته‌ها نیست.

۵. تابع معادل scheduler را در هسته لینوکس بیابید. جهت حفظ اعتبار اطلاعات جدول پردازه‌ها از قفل‌گذاری استفاده می‌شود. این قفل در لینوکس چه نام دارد؟

تابع schedule در لینوکس معادل این تابع است که در این لینک و این لینک یافت می‌شود.

از یک قفل به نام tasklist_lock در لینوکس استفاده می‌شود.

این قفل برای مثال در این خط قابل مشاهده است.