

# زمانبندی پروژه ها

امیر فراهانی  
نسا عباسی مقدم

# زمانبندی در xv6:

```
void
scheduler(void)
{
    struct proc *p;
    struct cpu *c = mycpu();
    c->proc = 0;

    for(;;){
        // Enable interrupts on this processor.
        sti();

        // Loop over process table looking for process to run.
        acquire(&ptable.lock);
        for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
            if(p->state != RUNNABLE)
                continue;

            // Switch to chosen process. It is the process's job
            // to release ptable.lock and then reacquire it
            // before jumping back to us.
            c->proc = p;
            switchvm(p);
            p->state = RUNNING;

            swtch(&(c->scheduler), p->context);
            switchkvm();

            // Process is done running for now.
            // It should have changed its p->state before coming back.
            c->proc = 0;
        }
        release(&ptable.lock);
    }
}
```

زمان بندی Round Robin

تابع scheduler از فایل proc.c

یک context switch صورت میگیرد  
پوینتر c نشان دهنده پردازنده است  
پردازه بعدی که با p مشخص شده، زمانبندی میشود  
در یک کوانتوم زمانی پردازنده دست پردازه است

```
swtch (&(c->scheduler), p->context);
```

# زمانبندی در xv6:

اگر کار پردازش تمام شود، از صف خارج میشود  
ولی ممکن است کارش تمام نشود.  
Timer interrupt یک وقفه صادر میکند. (trap.c)

```
if(myproc() && myproc()->state == RUNNING &&  
    tf->trapno == T_IRQ0+IRQ_TIMER)  
    yield();
```

تابع yield از فایل proc.c صدا زده می شود  
پردازش به ته صف میرود و پردازنده به اولین پردازش سر صف تعلق میگیرد.

## زمانبندی در xv6:

```
void
yield(void)
{
    acquire(&ptable.lock);
    myproc()->state = RUNNABLE;
    sched();
    release(&ptable.lock);
}
```

تابع yield از proc.c

- گرفتن قفل بروری پراسس ها
- تغییر پراسس انتخاب شده به حالت قابل اجرا
- اجرای تابع زمانبد
- رها کردن قفل روی پراسس ها

زمانبندی بازخوردی چند سطحی:

صف اول: Robin Round

صف دوم: LCFS

صف سوم: BJF

زمانبندی بازخوردی چند سطحی:

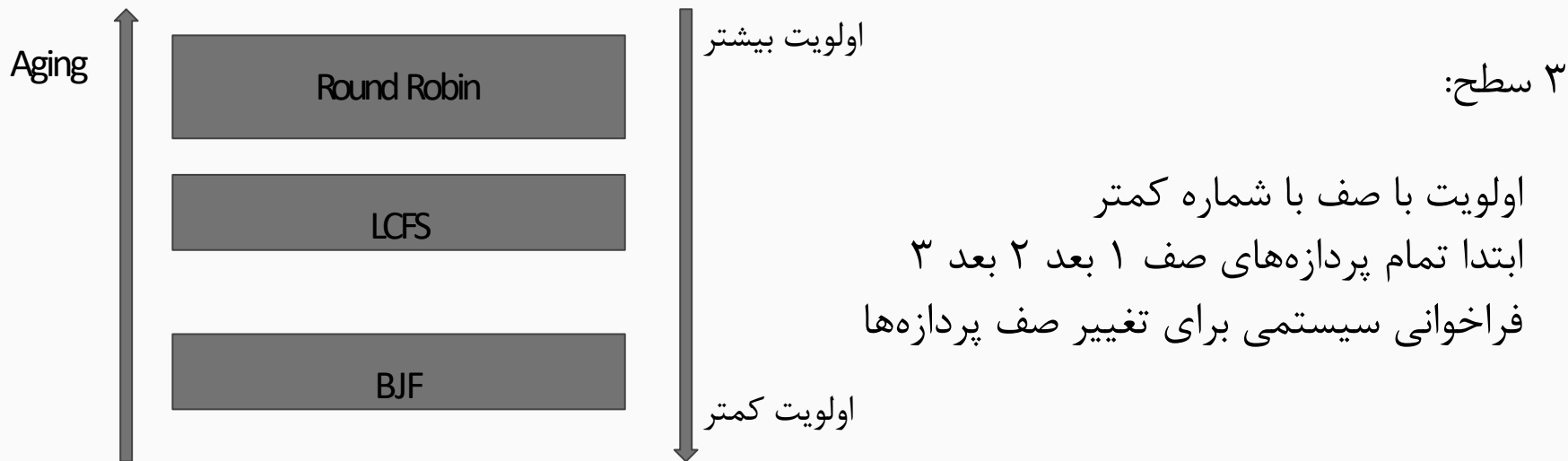
صف اول: Robin Round

صف دوم: LCFS

صف سوم: BJF

زمانبندی جدید

## زمانبندی بازخوردی چند سطحی:





## زمانبندی Round Robin:

- یک کوانتوم زمانی در نظر میگیریم.
- زمانبند پردازش را برای یک بازه حداکثر یک کوانتومی زمان بندی میکند
- اگر کمتر از یک کوانتوم کار پردازش طول بکشد، پردازش پردازنده را رها میکند و زمانبند پردازش بعدی را از سر صف انتخاب میکند.
- اگر بیشتر طول بکشد، یک اینترپت زمانی صادر میشود، پردازش در حال اجرا به ته صف می رود و زمانبند پردازش بعدی را از سر صف انتخاب می کند.

## زمانبندی LCFS:

- آخرین پردازهای که وارد صف شده اول اجرا می شود.
- برای پیاده سازی باید زمان ورود پردازها را نگهداری کنید.
- فایل c.proc و h.proc برای نگهداری این اطلاعات باید ویرایش شود.

## زمانبندی BJT:

- پردازهای که با توجه به فرمول بیشترین مقدار را داشته باشد اول انجام میشود

## مکانیزم Aging:

- در هر صفی هر پردازش ای ک<sup>۰</sup> بیشتر از ۸۰۰۰ سیکل زمانبندی نشد یک سطح بالاتر میرود.

$$۱ \leftarrow ۲ \leftarrow ۳$$

- جلوگیری از starvation

# فراخوانی های سیستمی:

- تغییر صف پردازش
- مقداردهی بلیت بخت آزمایی
- چاپ اطلاعات

## برنامه تست:

- برای تست یک برنامه سطح کاربر بنویسید.
- شامل تعدادی پردازش و عملیات پردازشی (به قدر کافی طولانی باشد)
- اجرا در پس زمینه: foo&