زمانبندی پردازه‌ها

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;
      switchuvm(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

زمانبندی در xv6:

یک context switch صورت می‌گیرد

پوینتر c نشان‌دهنده 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 پردازه 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 می‌شود
پردازه به ته صف میرود و پردازنده به اولین پردازه 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

زمانبندی بازخوردی چند سطحی:

صف اول: Round Robin

صف دوم: LCFS

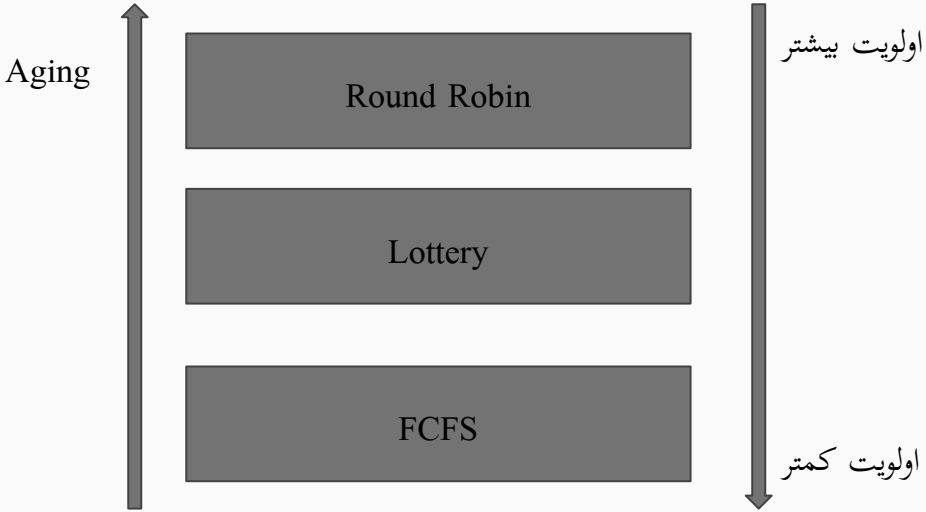
صف سوم: MHRRN

زمان‌بندی جدید

زمانبندی بازخوردی چند سطحی:

- صف اول: Round Robin
- صف دوم: زمان‌بندی بخت‌آزمایی
- صف سوم: FCFS

زمانبندی بازخوردی چند سطحی:



۳ سطح:

اولویت با صف با شماره کمتر

ابتدا تمام پردازه‌های صف ۱ بعد ۲ بعد ۳

فراخوانی سیستمی برای تغییر صف پردازه‌ها

زمانبندی Round Robin:

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

زمانبندی بخت‌آزمایی:

- هر پردازه با توجه به تعداد بلیت شانشی که دارد احتمال انتخاب شدنش به عنوان پردازه بعدی برای اجرا مشخص می‌شود

- هر پردازه تعدادی بلیت شانس دارد و پردازنده به صورت تصادفی یک بلیت را انتخاب نموده و پردازه صاحب آن بلیت، اجرا خواهد شد

زمانبندی FCFS:

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

مکانیزم Aging:

- در هر صفی هر پردازه ای که بیشتر از ۸۰۰۰ سیکل زمانبندی نشد یک سطح بالاتر میرود.

  3 ← 2 ← 1

- جلوگیری از starvation

فراخوانی‌های سیستمی:

- تغییر صف پردازه
- مقداردهی بلیت بخت‌آزمایی
- چاپ اطلاعات

برنامه تست:

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