به نام خدا

پروژه پایان ترم درس سیستم های عامل   
حسین خادم 400521279

سید حامد سادات 400521405

تابع scheduler بعد از اجرای سیستم عامل در هر هسته اجرا شده و یک لوپ بی نهایت است و مسئولیت آن انتخاب پروسس های runnable و اجرای آنها بعد از آوردن دیتای مورد نیاز پروسس از حافظه است و بعد از اتمام کار پروسس انتقال کنترل به این تابع کار خود را ادامه میدهد Ptable هم یک struct است که در آن دو متغیر وجود دارد:

1\_یک mutex lock برای قفل کردن جدول که مانع از مشکلات مربوط به همگام سازی شود

2\_لیست پروسس های قرار داده شده در زمانبند

زمانبند در XV6 در موقعیت های زیر اجرا می شود :

1.هنگام راه اندازی اولیه سیستم => زمانبند xv6 در طول فرآیند بوت سیستم راه‌اندازی می‌شود. این در تابع main() در فایل main.c اتفاق می‌افتد که پس از تنظیمات اولیه سیستم، شامل مقداردهی ساختارهای جدول فرآیند و راه‌اندازی فرآیند اولیه (initproc)، تابع scheduler() را فراخوانی می‌کند.

2.موقع context switching :

مانبند در هر موقعیتی که نیاز به تعویض زمینه بین فرآیندها باشد، اجرا می‌شود. این معمولاً در موارد زیر اتفاق می‌افتد:

* هنگامی که یک فرآیند به حالت خواب (sleep) می‌رود.
* هنگامی که یک فرآیند به پایان می‌رسد.
* هنگامی که یک فرآیند داوطلبانه CPU را آزاد می‌کند.
* زمانی که Interrupt تایمر رخ می‌دهد و زمان تخصیص داده شده به فرآیند فعلی به پایان می‌رسد.

فرآیند تحت wait() و Timer Interrupt و exit() خارج می شود.

در سیستم‌عامل xv6، تابع wait زمانی اجرا می‌شود که یک فرآیند والد بخواهد منتظر بماند تا یکی از فرآیندهای فرزندش به پایان برسد. این تابع به فرآیند والد اجازه می‌دهد که تا پایان اجرای فرآیند فرزند به حالت انتظار بماند و سپس منابع مربوط به آن فرزند را آزاد کند.

هنگامی تابع exit() اجرا می شود که یک فرآیند انجامش به اتمام برسد.

Timer Interrupt هم هنگامی انجام می شود که سخت افزار تایمر Overflow کند. آن موقع فلگ اینتراپت مربوطه روشن شده و اینتراپت رخ میدهد

زمان در یک کامپیوتر توسط سخت افزار تایمر محاسبه میشود  
برای همین ما برای برش های زمانی هم میتوانیم از آن استفاده کنیم اما منتظر تایمر ماندن و polling وقت سیستم را هدر میدهد  
برای همین سیستم عامل xv6 سیستم tick استفاده میکند

به این صورت است که تایمر بر یک بازه ثابت تنظیم شده و هربار که به مقدار مورد نظر برسد سیگنال اینتراپت را داده و مقدار تیک را افزایش میدهیم و اگر پروسسی در حال اجرا باشد،cpu را ازش میگیریم و دوباره schedule میکنیم  
در سیستم xv6 هر پروسس به اندازه 1 tick کار میکند   
ما میتوانیم با وارد کردن یک متغیر میزان تیک مورد نیاز را کنترل کنیم  
اما ما با وجود امکان کار با متغیر برای شمارش تیک ها و and کردن آن با شرط مربوطه برای yield کردن،ما روش دیگری رفته و تعداد تیک ها را داخل تابع yield کنترل میکنیم

برنامه ایی در کد نوشته شده که تعدادی پروسس درست کرده و هر یک از آنها را با یک تسک نسبتا سنگین ران میکنیم

تعداد پروسس ها و میزان سنگینی پروسس ها را تغییر داده و تعداد context switch را نمایش میدهیم

بعد از تست کردن حالت های مختلف،به این نتیجه رسیدیم که تحت پروسس هایی با که زمان اجرای کوتاه تر،تفاوت RR با ERR خیلی جزئی بوده در حد 7 یا 8 تا تفاوت داشتند

در پروسس هایی با زمان اجرای متوسط هم این تفاوت به 15 تا 18 تا CS میرسید

اما وقتی پروسس ها را از یه حدی سنگینتر میکنیم تفاوت زیادی مشاهده میکنیم

در آخرین اجرا تحت کد داخل userapp کد اجرایی ما 65 context switch داشت اما تحت زمانبند عادی xv6 ما به طور میانگین 254 تا context switch داشتیم  
به طور کلی کاهش زمان CS زمان اجرای برنامه را بهبود میبخشد