علی اطهری
پرهام سلطانی

پروژه پایانی سیستم های عامل

تابع scheduler پیشفرض

تابع scheduler به کمک شمارنده های runable_count و runable_count_check تعداد فرآیند های آماده به اجرای سیستم را بدست آورده و طبق آن زمان بندی را تنظیم میکند. در ابتدا با فراخوانی sti() وقفه ها را فعال کرده، سپس با یک حلقه روی تمام فرآیندها ، فرآیندهای آماده اجرا را پیدا میکند. با افزایش شمارنده context_switches فرآیند را به حالت RUNNING تغییر داده و در پردازنده اجرا میکند. Swtch متن context را تغییر و switchkvm بعد از اجرای فرآیند، پردازنده را به حالت اصلی برمیگرداند. شمارش تعداد فرآیندهای آماده به اجرا runable_count_check مجددا انجام شده و زمانبندی تایمر با timer_intervals طبیق داده میشود. نهایتا قفل آزاد شده و تایمر lapic تنظیم میشود.
تابع scheduler_rr از الگوریتم برنامه‌ریزی Round Robin (چرخشی) استفاده می‌کند.مشابه تابع قبلی، حلقه بی‌نهایت دارد و با فعال‌سازی وقفه‌ها با دستور sti() شروع می‌کند. باقی فرآیند بطور کلی شبیه تابع قبلیست.

نحوه اجرای زمان بندی و Timer

در تابع setlapictimer مقدار 10000000 به رجیستر TICR نوشته میشود. این مقدار تعیین میکند که lapic timer چندبار قبل از ایجاد یک وقفه تایمر شمارش کند. تابع lapicw مقداز value را به آدرس محاسبه شده توسط lapic + index مینویسد، جایی که lapic اشاره گر به پایه آدرس رجیسترهای apic است. TICR مقدار اولیه تایمر را تعیین میکند، TCCR مقدار فعلی تایمر را نشان میدهد، TDCR نرخ تقسیم تایمر را تنظیم میکند و LVT Timer برای تنظیم نوع وقفه تایمر استفاده میشود.

عملکرد Lapic

متغیر lapic یک پوینتر به پایه آدرس رجیسترهای lapic است و در فایل mp.c مقداردهی اولیه میشود.
تابع lapicinit رجیستر SVR را با مقدار Enable تنظیم میکند، سپس رجیسترهای LINT0 و LINT1 را با مقدار MASKED تنظیم میکند.همچنین نرخ تقسیم تایمر TDCR را به X1 تایمر را به حالت دوره ای و IRQ_Timer و مقدار اولیه TICR را به 10000000 تنظیم میکند. اگر نسخه APIC حداقل 4 باشد ، رجیستر PCINT را با مقدار MASKED تنظیم میکند. رجیستر ERROR را به وقفه IRQ_ERROR نگاشت میکند. نهایتا مقدار رجیستر ESR را به صفر تغییر میدهد و مقدار صفر را به رجیستر EOI نیز مینویسد تا به ترتیب رجیسترهای وضعیت خطا را پاک کرده و سپس وقفه های معلق را تایید کند.
تابع lapicid شناسه apic را برمیگرداند. اگر بطور محلی مقداردهی نشده باشد، مقدار صفر را برمیگرداند.همچنین Lapiceoi برای ack یک وقفه استفاده میشود. تابع lapicstartap یک پردازنده اضافی را با ارسال وقفه های INIT و STARTUP به آدرس ورودی مشخص شروع میکند.

نحوه افزایش طول بازه زمان بندی

از آن جایی که Timer طول زمان بازی بندی را مشخص میکند. برای تغییر طول بازه زمان بندی نیاز بود تا رجیستر Timer در LAPIC را تغییر دهیم. برای این کار یک تابع در فایل lapic.c نوشتیم که یه عدد به عنوان ورودی میگیرد و مقدار Timer را برابر آن عدد قرار میدهد.
برای آن که بتوانیم از این تابع در تابع scheduler یا فایل های دیگر استفاده کنیم نیاز بود تا این تابع را در فایل defs.h تعریف کنیم.
با این کار میتوانیم در تابع scheduler طول بازه زمان بندی را بر اساس نیاز تغییر دهیم.

در تابع scheduler هنگام افزایش بازه زمانی دقت شده تا اندازه آن بیشتر از 2 به توان 32 نشود. زیرا این یک رجیستر 32 بیتی هست و در غیر این صورت overflow رخ می دهد. برای همین اندازه بازه زمان بندی تا یه حدی دو برابر میشود و بعد از آن ثابت میماند.

عملکرد Tick

یکی از روش های زمان بندی استفاده از تایمرهای دوره ای یا Tick است و هربار که یک تایمر منقضی میشود، یک وقفه ایجاد میکند. به کمک یک متغیر شمارنده میتوان تعداد Tick ها را ذخیر کرد. هربار که وقفه تایمر رخ میدهد، شمارنده را یک واحد افزایش میدهیم و مقدار آن را در وقفه تایمر خروجی میدهیم.
متغیر را در فایل defs.h بصورت خارجی extern uint ticks تعریف و همچنین در lapic.c نیز با مقدار صفر initialize میکنیم. سپس درون تابع trap ، شمارنده tick را بروزرسانی و IRQ_TIMER را مدیریت میکنیم.

```
void trap(struct trapframe *tf)
{
        if(tf->trapno == T_IRQ0 + IRQ_TIMER){
                if(cpuid() == 0){
                        ticks++;
                        cprintf("Tick: %d\n", ticks);
                        wakeup(&ticks);
                }
                lapiceoi();
}
```

تست و ارزیابی تغییرات

برای تست عملکرد زمانبندی جدید یک فایل به نام schedulertest نوشتیم و آن را به Makefile اضافه کردیم که بتوان آن را اجرا کرد. هنگام اجرا این برنامه 50 تا فرایند فرزند تولید میشود و همزمان هر کدام در یک لوپ مشغول به انجام محاسبات میشوند.
برای اندازه گیری تعداد context switch ها رخ داده یک متغیر گلوبال به نام context_switches تعریف کردیم و در تابع scheduler هر دفعه که یک فرایند را اجرا می شود افزایش می یابد. سپس یک تابع به نام sys_getcontextswitches تعریف کردیم تا بتوان مقدار این متغیر را در فایل های دیگر به دست آورد.
فایل schedulertest عملیات تست را 20 بار انجام میدهد و میانگین تعداد context switch و زمان اجرا را محاسبه میکند.

	RR	New scheduler
میانگین context switch ها	66.750	64.250
میانگین زمان اجرا	16.750	14.250

همانطور که در نتایج معلوم است با استفاده از زمانبندی جدید ۳.۷ درصد کاهش تعداد context switch داشته و ۱۵ درصد کاهش زمان اجرا داشته. در کل الگوریتم زمانبندی جدید در این تست بهتر عمل کرده.