OPERATING SYSTEM LAB

SESSION 3

SeyedMehdi HajiSeyedHossein	810100118
Alireza Hosseini	810100125
AmirAli Shahriary	810100173

سوال ۱ : چرا فراخوانی تابع ()sched منجر به فراخوانی تابع ()scheduler می‌شود؟

هر پردازه ای که می خواهد پردازنده را رها کند باید lock های دیگر را رها کند و استیت خودش را آپدیت بکند . با فراخوانی ()sched شروط دوباره بررسی شده هر تابع با فراخوانی scheduler باعث آغاز کار زمان بند هسته می شود و در تابع فوق به دنبال پردازه قابل اجرا می گردد و در صورت یافتن RUNNABLE process با کمک تابع swtch تعویض متن صورت می گیرد و رجیستر های قدیمی در آدرس مربوطه آن context ذخیره می شوند . یعنی عملا context switch رخ می دهد (یعنی context فعلی در proc->context ذخیره شده و context زمانبند را در cpu->scheduler اجرا کند) و سپس به scheduler سوییچ می کنیم و scheduler جایگزین پردازه فعلی شده و حالت آن به RUNNING process تغییر می کند.

سوال ۲ : صف پردازه هایی که تنها منبعی که برای اجرا کم دارند پردازنده است، صف آماده یا صف اجرا نام دارد. در xv6 صف آماده مجزا وجود نداشته و از صف پردازه ها بدین منظور استفاده میگردد. در زمانبند کاملاً منصف در لینوکس، صف اجرا چه ساختاری دارد؟

زمان بند کاملا منصف در لینوکس بنا به صف اجرا در لینوکس توسط یک درخت قرمز-سیاه پیاده سازی می گردد که در چپ ترین گره پردازه ای جای می گیرد که کمترین برش زمانی را در حین اجرا دارد. در لینوکس vruntime بار اولویت بندی استفاده شده که بیانگر ترکیب وزن و زمان اجرا آن است که به عنوان کلید بیان شده و در درخت قرمز سیاه چپ ترین گره حامل کوچکترین مقدار vruntime است.

سوال ۳ : همانطور که در پروژه اول 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 یا مجزا بودن صف های زمانبندی بررسی نمایید. و یک مزیت و یک نقص صف مشترک نسبت به صف مجزا را بیان کنید.

ساختار کلی لینوکس بدین صورت است که از صف های زمان بندی مجزا استفاده می کند با این تفاوت که در سیستم عامل xv6 از یک صف به صورت مشترک استفاده می گردد.

مزیت : در حالتی که صف ها مجزا باشند نیاز به هندل کردن لود آنها هستیم و مکانیزمی مانند load balancing وجود دارد که در صف های مشترک نیازی به آن نداریم . همچنین پیاده‌سازی صف مشترک ساده تر است.

نقص : در صف مشترک ممکن است مشکلاتی در دسترسی همزمان به صف ایجاد بشود که باید آنرا کنترل کنیم در صورتی که به تبع نوع پیاده سازی صف های مجزا این اتفاق رخ نمی دهد.

سوال ۴ : در هر اجرای حلقه، ابتدا برای مدتی وقفه فعال میگردد. علت چیست؟ آیا در سیستم تک هسته ای به آن نیاز است؟

چنانچه پردازه قابل اجرایی نداشته باشیم مثلا هنگامی که همه پردازه ها منتظر عملیات IO هستند و عملا هیچ پردازه RUNNABLE ای نداریم و اگر زمانبند وقفه‌ها را همیشه غیرفعال می‌گذاشت عملیات IO هیچگاه انجام نمی‌پذیرفت و به همین دلیل در ابتدای حلقه برای مدتی فعال می‌گردد که این مشکل پیش نیاید. جواب بخش بعدی نیز بله است زیرا که اتفاق فوق ممکن است رخ دهد و این فریز شدن سیستم ربطی به هسته‌های سیستم ندارد.

سوال ۵ : وقفه‌ها اولویت بالاتری نسبت به پردازه‌ها دارند. به طور کلی مدیریت وقفه‌ها در لینوکس در دو سطح صورت میگیرد. آنها را نام برده و به اختصار توضیح دهید. اولویت این دو سطح مدیریت نسبت به هم و نسبت به پردازه‌ها چگونه است؟

لینوکس مدیریت وقفه را به دو نیمه تقسیم بندی کرده است : نیمه بالایی که به آن FLIH می گویند و نیمه پایینی که SLIH نامیده می شود. درنیمه بالایی که یک سرویس استاندارد مدیریت وقفه ها است که در آن رخداد وقفه های دیگر با همان شماره غیر فعال است و وظیفه آن مدیریت وقفه های ضروری در کمترین زمان ممکن است همچنین آنها باعث چشم پوشی کردن از وقفه ها می شوند . مدیریت وقفه در نیمه پایینی بدین شیوه است که وقفه ها غیرفعال نیستند ولی یک زمانبندی موجود است که سبب می شود هیچ وقفه نیمه پایینی باعث وقفه در خود نشود در کل وظیفه بخش هایی از پردازش وقفه را بر عهده دارد که زمان بر هستند و این کار به مانند یک پردازه انجام می پذیرد آنها در یک صف اجرا قرار می گیرند و منتظر پردازه میماند بعلت مشکلات احتمالی زمان طولانی برای اجرایشان غالبا به مانند پردازه ها و ریسه ها زمانبندی می شوند.

تقسیم بندی فوق سبب می شود هر محاسبه پیچیده را در یک جواب به یک وقفه انجام داده و بدون نگرانی از اختلال خودش توسط وقفه دیگری باشد همچنین این ساختار سبب می شود وقتی وقفه ها در نیمه پایینی هستند و هسته در شرف عملیات بحرانی است نیمه پایینی غیرفعال شده و از وقفه در عملیان پیشگیری می کند و پس از آن عملیات می تواند مجدد آنها را فعال کند و وقفه ها در نیمه پایینی ایجاد شوند.

مدیریت وقفه ها در صورتی که بیش از حد زمانبر شود، میتواند منجر به گرسنگی پردازه ها گردد. این میتواند به خصوص در سیستم های بیدرنگ مشکل ساز باشد. چگونه این مشکل حل شده 16 است؟

چنانچه پردازه ای به علت داشتن اولویت کمتر نسبت به باقی پردازه ها مدت غیر مشخصی را در صف آماده سپری کند دچار گرسنگی پردازه شده است. برای رفع آن راه حل Aging توصیه شده است که عملا اولویت‌ها به مرور زمان افزایش پیدا می‌کنند .

با توجه به توضیحات بخش قبلی برای جلوگیری از گرسنگی وقفه‌ها آن وقفه زمان بر را به نیمه پایینی انتقال می‌دهند.