موضوع «پویان حسابی - امیر ماهیار» «پروژه سیستم عامل تست اول» تاریخ / /

1) در Proc.h یک struct قرار دارد و ابتدا procstate را از جنس enum تعریف کرده تا state فرآیند را نگه داشته باشیم. در struct proc گفته شده چه چیزهایی در process موجود است:

uint sz; سایز پراسس را نگه می‌دارد در یک int بدون علامت

char* kstack; در kernel stack مواردی وجود دارد و این پوینتر به آدرس آن است.

enum procstate {Unused, EMBRYO (new), Sleeping (Wait), Runable (Ready), Running, Zombie (terminated)};

pid و id های پراسس والد و فرزند که مشخص است و یک پوینتر به آن ها نیاز داریم.

trapframe *tf در kernel stack یک trapframe داریم تا اطلاعات پراسس را نگه دارد و این به آن اشاره می‌کند.

موضوع .... پراسس‌ها و رجیسترهایی تاریخ / /

وقتی بین پراسس‌ها Context switch انجام می‌شود نیاز است اطلاعات ۷ باشد. Context *context:

وقتی پراسس sleeping هست و می‌خواهد آماده شود باید از آدرسی شروع کند که در آن قرار می‌گیرد. void* chan:

اگر ۰ نباشد پراسس kill شده و دیگری اسم آن است. int killed, char name:

آرایه‌ای از فایل‌های باز شده که آدرس آنها را برای افزایش پرفورمنس نگه می‌دارد. file *ofile

در struct context: اعداد طبیعی که رجیسترهایی هستند در kernel برای Context switch که مقدار آن‌ها

می‌باشد. آنها در بالای stack ذخیره می‌شوند که آنها را توصیف می‌کند Constant

در struct CPU: برای هر CPU این struct نگهداری می‌شود تا عملیات‌های لازم را انجام دهد.

context *scheduler برای context switch است که وارد زمان‌بندی می‌شود

task_state *ts برای وقتی‌ها نیاز داریم که stack های آن‌ها را بسازیم

volatile booted یک عدد منتقل می‌کند CPU شروع شده است یا خیر (بوت شده)

① ادامه در proc.c:

myproc و mycpu: که پراسس فعلی و پردازنده‌ی فعلی را می‌دهد.

allocproc: زمانی که فرایندی ساخته می‌شود این تابع فراخوانی می‌شود و آنرا در حافظه allocate کرده و مقداردهی

fork: پراسس جدیدی می‌سازد که فرزند آن است. اگر حافظه‌ی آن ممکن نشود یعنی نتوانست بسازد. اگر هم شد یعنی موفقیت

آمیز بود. اگر بیشتر از صفر شود برای فرایندی است که والد را استفاده می‌کنیم.

exit: پراسس فعلی را می‌بندد و در حالت Zombie می‌برد تا پراسس والد آن را wait فراخوانی کند.

wait: برای بستن پراسس‌های فرزند استفاده می‌شود که در مدت چهارم برای نگهداری زمان‌های مورد نیاز

تابع مشابه آن می‌نویسیم:

Scheduler: برای زمان‌بندی پردازنده استفاده می‌شود که تشکیل صف از پراسس‌های ready می‌باشد. برای پیاده‌سازی الگوریتم‌های زمان‌بندی باید از این ساختار استفاده کنیم.

yield: این تابع برای متوقف کردن فرایندی که در حال اجراست و زمان آن تمام نشده استفاده می‌شود.

(۴) الگوریتم زمان بندی آن Round Raubin می باشد. در فایل proc.c یک تابع به نام scheduler وجود دارد که در آن یک حلقه دائمی است که به ترتیب از ساده پراسس ها این شروع می کند و در حال آنکه Ready است می گردد اگر Ready بود آنرا اجرا می کند سپس CPU را از او می گیرد، اطلاعات kernal را load می کند و Context ها را انجام می دهد. بر اساس عملکرد آن متوقف شدنی (Preemptive) است یعنی trap می آید و در زمان خاصی پراسس را متوقف می کند. در trap یک شرط داریم که trap ها در رجیستر timer می آید، در آن شرط شرطی دیگر وجود دارد که اگر پراسسی در حال حاضر وجود داشته باشد و در حال اجرا باشد و trap آن timer باشد، CPU را از پراسس می گیرد (آنرا Ready می کند)

فایل ها: proc.c - trap.c - defs.h و scheduler را فراخوانی می کند

۴) در فایل syscall.c یک تابع به نام syscall وجود دارد که فراخوانی سیستمی را انجام می دهد.

در syscall.h تمام system call ها و شماره‌هایی که به آن‌ها اختصاص داده شده قرار دارد.

برای فراخوانی سیستمی ابتدا پرامتر را می‌برد سپس شماره فراخوانی سیستمی که در حال حاضر در trapframe پرامتر ما قرار دارد را می‌گیرد.

چک می‌کند که این شماره فعال باشد و یک syscall آن موجود باشد، آن فراخوانی سیستم را اجرا می‌کند و نتیجه آن را در رجیستر eax از trapframe پرامتر می‌ریزد. اگر نبود هم مقدار (-1) در آن می‌ریزد.

برای تعریف فراخوانی سیستمی جدید ابتدا در syscall.h شماره‌ای به آن نسبت می‌دهیم، سپس در sysproc.c تابع مورد نظر را می‌نویسیم و بعد در syscall.c آن را extern کرده و در function *array آن را تعریف

ابتدا در usys.S یک system call را می نویسیم
می کنیم. تا اینجا در سطح کرنل انجام شده است برای اینکه در سطح کاربر بتوانیم از user.h می رویم و آن تابع
را تعریف می کنیم تا کاربر interface آنرا داشته باشد. در نهایت چون ما فایل sysproc.c تابع مورد نظر را
نوشتیم ولی برنامه تابع را proc.c بود ولی آنرا include نکرده بودیم پایین defs.h include شده است
آنرا تعریف کنیم.