«به نام خدا»

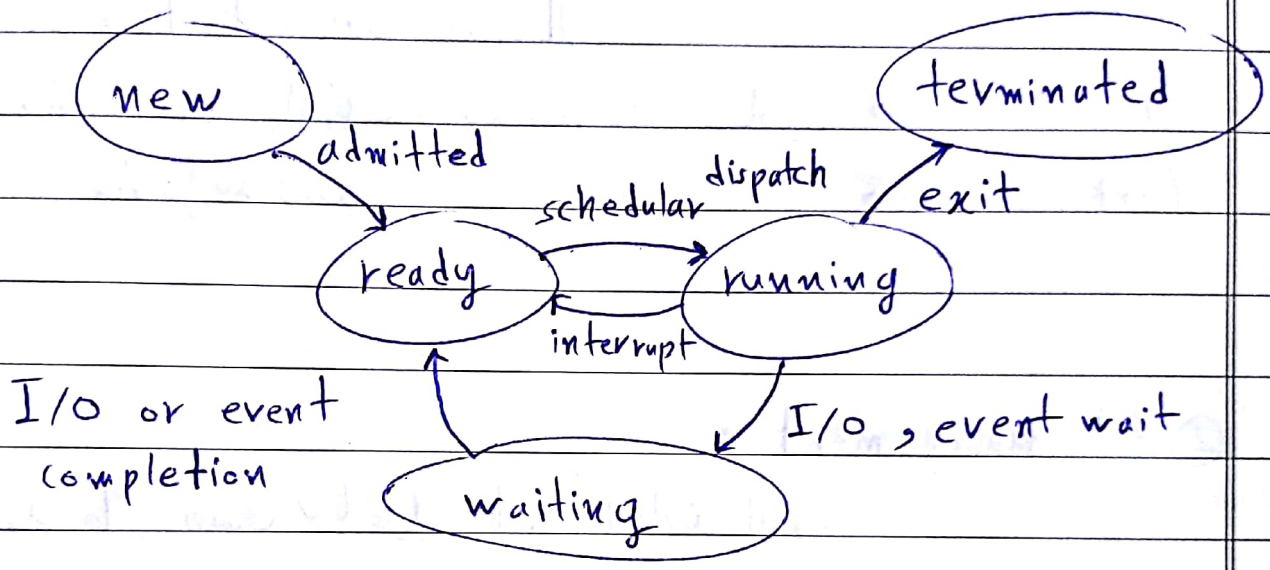
امیرمحمد پیرحسینلو

9531014

تمرین سوم OS

① فرآیند برنامه‌ای است که memory بارگذاری شده و در حال اجرا است.

states: new , ready , running , waiting , terminated



② system call ها به عنوان رابط هایی هستند که با استفاده از آن ها می توانیم از سرویس های آماده ی سیستم عامل (برنامه های از قبل نوشته شده) استفاده کنیم.

① process control: فراخوانی هایی است که برای مدیریت و انجام عملیات های روی فرایند ها به کار می رود. برای مثال:

abort() ⟶ پایان دادن به اجرای یک فرآیند به صورت غیرعادی

fork() ⟶ ایجاد فرآیند فرزند

② file management:

فراخوانی هایی هستند که با فایل ها سروکار دارند مانند:

create() ⟶ ساختن فایل

read() ⟶ خواندن از فایل

③ device management:

برای اختصاص دادن منابع فیزیکی و مجازی مانند memory و I/O و فایل و... به فرآیندها از فراخوانی های این مدل استفاده می کنیم. مانند فراخوانی های request() و release().

پس از اتمام کار، منابع را به سیستم عامل برمی گردانیم.

④ information maintenance:

فراخوانی هایی وجود دارد که برای انتقال اطلاعات میان برنامه ی کاربر و سیستم عامل به کار می رود مانند time() که زمان را برمی گرداند و یا

dump() که از memory، snapshot می‌گیرد و برای debugging به کار می‌رود.

⑤ communication:

دو نوع روش برای برقراری ارتباط میان فرآیندها وجود دارد که شامل موارد زیر است:

Ⓐ message passing Ⓑ shared memory

در message passing، فرآیندها برای برقراری ارتباط از فراخوان‌های سیستمی مانند open-connection()، get_processid() و ... استفاده می‌کنند.

⑥ protection:

فراخوانی‌هایی که برای تأمین امنیت سیستم به کار برده می‌شود مانند set_premission() که اجازه‌ی دسترسی به منابع مانند فایل (ها) را به کاربران می‌دهد.

③ مزیت: تغییرات در هر لایه تنها در آن لایه اعمال می‌گردد و نیاز به ایجاد تغییر در لایه‌های دیگر نیست (البته به شرطی که interface با لایه‌ی پایین و بالا تغییر نکند.)
راحتی ساخت و debugging (عیب‌یابی)
لایه‌های دیگر نیاز ندارند که بدانند لایه‌ی دیگر چگونه پیاده‌سازی شده است.

معایب: ① تعریف وظایف هر لایه سخت است. ② برنامه‌ی کاربر برای استفاده از سرویس‌های سیستم عامل باید چندین لایه را طی کند که باعث کاهش کارایی و افزایش تأخیر می‌شود. ③ بعضی از برنامه‌ها ممکن است بخواهند به‌طور مستقیم با لایه‌های پایین‌تر صحبت کنند که در این مدل تعریف نشده است.

④ بعضی از device ها مانند cellular phone ها دیسک سخت ندارند و برای نگهداری برنامه‌ی سیستم عامل باید آن را در firmware نگه‌داری کنند.

(5)

mechanism ← تعیین می‌کند که چگونه و از چه روش عملیاتی انجام شود.

policy ← نحوه پیاده‌سازی عملیات را نشان می‌دهد.
خود
پیاده‌سازی شده

جدایی این دو به خاطر انعطاف پذیری مهم است. معمولاً تغییرات در policy ممکن است موجب تغییرات در mechanism شود در صورتی که این مطلوب نیست. مکانیزمی خوب است که در برابر policy های گوناگون منعطف باشد و کار کند.

⑥ command interpreter : برنامه‌ای است که کاربر از طریق آن با OS interaction انجام می‌دهد. کاربر دستورات خود را به آن می‌دهد و آن نیز دستورات را به سیستم عامل می‌دهد تا اجرا شود.

system daemons : سرویس‌هایی که به وسیله‌ی برنامه‌های سیستمی که در زمان boot شدن بارگذاری می‌شوند و همیشه وقتی که kernel در حال اجرا است آن‌ها هم در حال اجرا هستند.

system services : محیطی ساده و راحت برای توسعه‌ی برنامه‌ها و اجرای آن‌ها فراهم می‌کند. مانند سرویس File management

bootstrap program : قطعه کدی است که kernel سیستم عامل را پیدا می‌کند.

API : مجموعه‌ای از function ها است که ورودی (پارامترها)، خروجی آن‌ها مشخص است و در اختیار برنامه‌نویس قرار می‌گیرد تا برنامه‌های خود را توسعه دهد. مثلاً POSIX برای سیستم عامل‌های UNIX و Linux