سیستم‌های عامل - پاسخنامه تکلیف اول

سوال ۱.

الف) در یک پردازنده چند هسته‌ای همه هسته‌های پردازشی یا پروسسورها روی یک چیپ یا سیستم کامپیوتری قرار دارند. در حالی که سیستم‌های clustered میتوانند شامل تعداد زیادی سیستمهای مالتی پروسسور یا معمولی باشند که از طریق ارتباطی مانند شبکه به یکدیگر متصل هستند.

ب) API call system در فضای کاربر استفاده میشوند و رابط call system یی هستند که در فضای کرنل جهت اجرای وظایف اصلی سیستم عامل اجرا می‌شوند و به برنامه های سطح کاربر جهت استفاده از منابع سخت افزاری سرویس میدهند. کار کردن با هاAPI راحت‌تر است و همچنین کمک می‌کند تا برنامه‌های نوشته شده را بتوان در سیستم‌های مختلف اجرا کرد، مادامی که از هایAPI آن پشتیبانی شود.

ج) وظیفه بوت سیستم (که در رام قرار دارد یا به صورت UEFI در پارتیشن مشخصی از دیسک است) شروع به کار یک سیستم کامپیوتری و راه اندازی درایورهای اولیه و فراهم کردن امکان انجام بعضی تنظیمات سخت افزاری است. این برنامه، باعث شروع به کار یک بوت لودر یا بوت سیستم عامل میشود . بوت لودر که روی دیسک ذخیره میشود، پس از بوت سیستم امکان شروع اجرای سیستم عامل مورد نظر را می‌دهد.

سوال ۲.

الف) صحیح. مدیریت فضای مشترک بر عهدهٔ برنامه نویس خواهد بود و باید برای مشکلاتی که ممکن است پیش بیاید (مثل Race Condition) راه حل ارائه کند.

ب) غلط. این ساختار داده در فضای حافظهٔ پروسه نگهداری نمی‌شود بلکه در ساختمان‌داده‌های کرنل نگهداری می‌شود. جایی که از دسترسی user هاapplication خارج است.

سوال ۳.

در محاسبات ابری با استفاده از مجازی‌سازی، منابع یک سیستم بزرگ با چندین پردازندهٔ چند هسته‌ای و صدها گیگابایت حافظهٔ اصلی و چندین ترابایت حافظهٔ جانبی که به صورت توزیع‌شده به هم متصل هستند، در قالب VM های متنوع قابل ارایه به کاربران زیادی خواهد بود.

تجرید: آرایه منابع سخت‌افزاری یا نرم‌افزاری در قالب VM به طوری که کاربرها مستقیما درگیر اخذ منابع یا توزیع برنامه خود روی سیستم توزیع‌شده نیستند بلکه آنچه از منابع کلاد نیاز دارند به صورت یک ماشین مجازی (یا کانتینر) در اختیار کاربر قرار می‌گیرد و لایه‌های پایین سخت افزار یا نرم افزار از دید کاربر پنهان است حتی ممکن است منابعی که به یک کاربر داده می‌شود

به صورت فیزیکی روی یک سیستم نباشند. همچنین protection و امنیت از این طریق بهتر قابل مدیریت است زیرا هر VM مانند یک sandbox است.

مدیریت منابع: ماژول‌های نرم افزاری کلاد، مسئول تقسیم منابع با توجه به درخواست کاربران و سرویسی که خریداری کرده‌اند و همچنین تنظیم لود روی منابع کلاد هستند و همانطور که دسترسی‌های همزمان به منابع یا زمان‌بندی استفاده از منابع توسط سیستم عامل مدیریت می‌شود، این موارد در کلاد هم توسط لایه‌های نرم‌افزاری کلاد که نقش سیستم عامل را دارند انجام می‌شوند.

تفاوت‌ها: به طور کلی تفاوت در اینجا با یک سیستم شخصی این است که توزیع‌شدگی منابع وجود دارد و بنابراین بحث‌های همزمانی و IPC و مدیریت لود متفاوت از یک سیستم شخصی است.

سوال ۴.

ممکن است که دستورات اجرا شونده توسط پردازنده نیاز به دسترسی به حافظهٔ اصلی داشته باشند (و فقط حافظهٔ کش کفایت نکند). در اینصورت DMA و پردازنده نمی‌توانند به طور همزمان از باس متصل به حافظهٔ اصلی استفاده کنند و با هم تداخل پیدا می‌کنند.

در این حالت اولویت دسترسی به حافظه با DMA است تا بتواند انتقال بین بافر هاIO و حافظهٔ اصلی را به انجام برساند و اطلاعاتی از دست نرود. یک سیکل CPU می‌تواند منتظر بماند و بدون مشکل بعد از اتمام DMA اجرا شود.

سوال ۵.

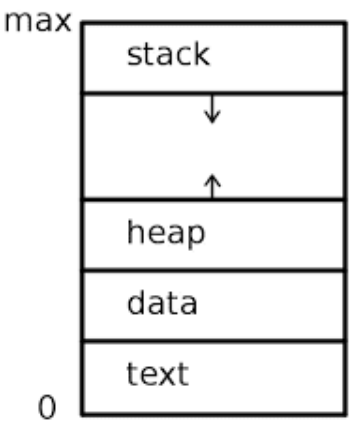
ساختار Monotonic از نظر کارایی بهتر و سریعتر از ساختار Modular است؛ اما توسعه آن سخت‌تر است و انعطاف پذیری کمتری دارد. به علت نزدیکی اجزای مختلف سیستم در ساختار Monotonic، سرعت آن بیشتر است. تغییر در یک بخش یا افزودن بخش جدید در ساختار Modular نیاز به تغییر کل سیستم ندارد و از نظر دیباگ برای توسعه دهنده آسان‌تر است.

سوال ۶.

یک برنامهٔ چند نخی (Multi-thread) را در نظر بگیرید که در نخ‌های مختلف آن مقدار یک متغیر گلوبال تغییر می‌کند. مثلا مقدار متغیر NumberOfUsers در نخ‌های مختلف آپدیت شوند. در این حالت ممکن است چند پردازنده به طور همزمان این چند نخ را اجرا کنند و بنابراین نخ‌ها کاملا همزمان ممکن است مقدار این متغیر را آپدیت کنند و خواندن و آپدیت متغیر منجر به ذخیره و آپدیت آن در کش محلی هر پردازنده به صورت جداگانه میشود. پس در صورت عدم اجرای یک همگام‌سازی، این مقادیر می‌توانند در کش‌ها متفاوت باشند.

سوال ۷.

الف)



شامل بخش های:

Text: کد برنامه

Data: متغیرهای سراسری

Stack: جهت اجرای برنامه (آرگومان‌های توابع، متغیرهای توابع و ...)

Heap: جهت تخصیص حافظهٔ پویا

ب) تعویض متن برای تغییر پروسهٔ در حال اجرا و قرار دادن پروسهٔ جدید روی پردازنده جهت اجرا انجام می‌شود .
PCB جهت ذخیرهٔ وضعیت پردازنده و پروسس استفاده می‌شود تا امکان اجرای ادامه پروسسی که فعلا از پردازنده خارج شده در زمان بعدی که زمان‌بندی می‌شود وجود داشته باشد.

ج) بدون اختیار: وقتی زمانبند به هر علتی (اتمام کوانتوم زمانی) پروسه را از روی پردازنده بردارد.
وقتی به دلیل نیاز به انجام عملیات IO ، پروسه پردازنده را رها می‌کند.

سوال ۸.

الف) مقدار ۵. چرا که هنگام fork، فضای حافظه که شامل داده‌ها است برای فرزند کپی می‌شود. پس تغییر در متغیر هم‌نام در والد و فرزند ارتباطی باهم ندارند. پس اینجا همان مقدار اولیه ۵ در فرزند می‌ماند.

ب) بله. هایfd باز شده توسط والد برای فرزندان باز می‌ماند.

ج) چون پروسهٔ والد بدون فراخوانی wait به اتمام می‌رسد، پروسهٔ فرزند یتیم می‌شود. در اینجا سیستم عامل این پروسهٔ یتیم را به init یا systemd (پروسهٔ اول هنگام شروع به کار سیستم عامل) می‌سپارد و init یا systemd سیستم کال wait را برای این پروسهٔ یتیم فراخوانی می‌کند. با اضافه‌کردن wait در انتهای پروسه والد (مثلا انتهای بلاک if مربوطه) می‌توان از یتیم‌شدن پروسس فرزند جلوگیری کنیم.

سوال ۹.

برنامه اصلی نقش shell را دارد. در ابتدای برنامه توسط Pipe یک پایپ ایجاد می‌شود. سپس با استفاده از fork یک پروسه فرزند برای shell ایجاد می‌شود که برای اجرای دستور اول استفاده می‌شود در این پروسس، با استفاده از dup۲ خروجی استاندارد و خطای استاندارد پروسس فرزند به سمت سر نویسنده پایپ هدایت می‌شود (یعنی از این پس به جای اینکه خروجی پروسس فرزند، در خروجی استاندارد نمایش داده شود در پایپ نوشته می‌شود). سپس به کمک دستور execv (از خانوادهٔ دستورهای exec،) دستور اول مورد نظر اجرا می‌شود و خروجی آن همانطور که بیان شد روی پایپ گذاشته می‌شود.

همچنین برنامه اصلی (shell) با fork دوم فرزند دومی برای اجرای دستور دوم می‌سازد. این پروسس فرزند جدید، با استفاده از dup۲ ورودی استاندارد را جایگزین خوانندهٔ پایپ می‌کند. یعنی پروسهٔ فرزندی که قرار است اجرا شود ورودیش را از پایپ برمی‌دارد. نهایتاً دستور دوم و آرگومان‌هایش باز هم به کمک execv اجرا شده است.

نهایتاً پروسهٔ والد این دو پروسه، با دوبار فراخوانی دستور wait به همراه ورودی null منتظر اتمام اجرای دو پروسهٔ اجرا کنندهٔ دو دستور می‌ماند. در اینجا null به این دلیل به کار رفته که ترتیب خاتمهٔ اجرا پروسه‌ها مهم نیست و پروسهٔ والد برای هر دو باید صبر کند.

دقت کنید که در این برنامه هر سری از پایپ که در پروسس مورد نظری استفاده نمی‌شود به درستی بسته می‌شود. پروسس والد هم هیچ استفاده‌ای از پایپ ندارد هر دو سر پایپ را می‌بندد.