

قسمت

اول

کلیات

OVERVIEW



سیستم عامل برنامه‌ای است که به عنوان واسطه مابین کاربر کامپیوتر و سخت افزار کامپیوتر، عمل می‌کند. هدف یک سیستم عامل آن است که محیطی ارائه دهد تا کاربر برنامه‌ها را به سهولت و سودمند اجرا نماید.

ما توسعه سیستم‌های عامل را از اولین سیستم‌های دست - بر تا چند برنامه‌ای و اشتراک - زمانی کنونی، تعقیب می‌کنیم. درک دلایل مربوطه به توسعه سیستم‌های عامل یک دیدگلی از این که سیستم عامل چه کار می‌کند و آن را چگونه انجام می‌دهد، برایمان ایجاد می‌نماید.

سیستم عامل با یستی کار صحیح سیستم کامپیوتری را تضمین نماید. جهت جلوگیری از تداخل برنامه‌های کاربر و عملکرد صحیح سیستم، سخت افزار با یستی مکانیزم مناسبی جهت تضمین رفتار درست سیستم ارائه دهد. ما معماری کامپیوتر پایه‌ای را توصیف می‌کنیم تا ما را در نوشتن یک سیستم عامل درست، یاری نماید.

سیستم عامل سرویس‌های مشخصی را برای برنامه‌ها و کاربران آن برنامه‌ها فراهم می‌سازد تا وظیفه برنامه‌نویسی را سهل‌تر نماید. سرویس‌های ویژه مزبور، از یک سیستم عامل به سیستم عامل دیگر، البته، متفاوت خواهند بود. اما انواع مشترکی از سرویس‌ها موجودند، که آنها را شناسایی نموده و غور خواهیم کرد.

فصل

۱

INTRODUCTION

مقدمه

(سیستم عامل^۱، برنامه‌ای است که به عنوان میانجی^۲ مابین کاربر کامپیوتر و سخت افزار کامپیوتر عمل می‌نماید. هدف سیستم عامل، ایجاد محیطی است که کاربر بتواند برنامه‌هایش را اجرا کند. هدف اولیه سیستم عامل این است که استفاده از سیستم کامپیوتر را سهل سازد. هدف بعدی آن است که استفاده از سخت افزار کامپیوتر را سودمند نماید.)

برای درک اینکه سیستم‌های عامل چه هستند، بایستی ابتدا، سیر تکامل و توسعه آنها را فهمید. در این فصل، سیستم‌های عامل را از اولین سیستم‌های دست - بر^۳ تا سیستم‌های چند برنامه‌ای^۴ اشتراک - زمانی^۵ امروزی، بررسی خواهیم نمود. در حین عبور از مراحل مختلف، خواهیم دید که چگونه اجزای سیستم‌های عامل به عنوان راه حل طبیعی برای مشکلات سیستم‌های کامپیوتری اولیه، تکامل می‌یابند. درک دلایل پیشرفت سیستم‌های عامل، پاسخ اینکه سیستم‌های عامل چه وظایفی را انجام می‌دهند و چگونه آنها را انجام می‌دهند، می‌باشد.

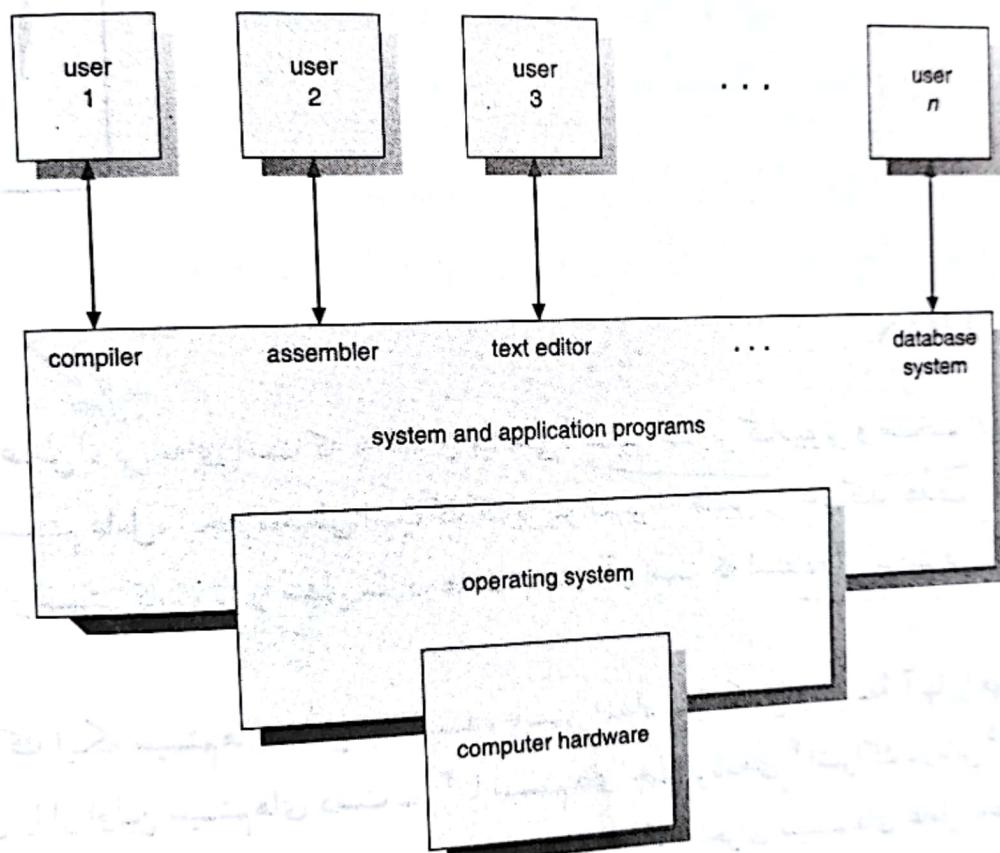
1. operating system
2. intermediary
3. hands-on system
4. multiprogram
5. time sharing

1.1. What Is an Operating System?

۱.۱. سیستم عامل چیست؟

سیستم عامل، یک بخش بسیار مهم در تقریباً تمام سیستم‌های کامپیوتروی می‌باشد. یک سیستم کامپیوتروی می‌تواند به چهار مؤلفه تقسیم شود: سخت‌افزار، سیستم عامل، برنامه‌های کاربردی، و کاربران (شکل ۱-۱).

4 Chapter 1 Introduction



شکل ۱-۱: دید انتزاعی از مؤلفه‌های سیستم کامپیوتروی

سخت‌افزار - واحد پردازشگر مرکزی (CPU)، حافظه، و وسایل ورودی/خروجی (I/O) - منابع محاسباتی پایه‌ای را تشکیل می‌دهند. برنامه‌های کاربردی - مانند واژه‌پردازها، کامپایلرهای سیستم‌های مدیریت بانک‌های

α

اطلاعاتی، بازیها، و برنامه‌های تجاری - روش‌هایی را بیان می‌کنند که این منابع مورد استفاده قرار می‌گیرند تا مسائل محاسباتی کاربران، حل شوند. کاربران متفاوت بسیاری (مردم، ماشین‌ها، کامپیوتروهای دیگر)، می‌توانند وجود داشته باشند که سعی در حل مسائل مختلف دارند. ولذا، برنامه‌های کاربردی متفاوت بسیاری هم می‌توانند وجود داشته باشند. سیستم عامل، استفاده از سخت‌افزار را مابین برنامه‌های کاربردی گوناگون کاربران، همانگ می‌سازد.

سیستم عامل، مشابه دولت^۱ می‌باشد. مؤلفه‌های یک سیستم کامپیوتروی، سخت‌افزار، نرم‌افزار و داده‌ها می‌باشند. سیستم عامل وسیله‌هایی برای کاربرد درست این منابع در عملکرد سیستم کامپیوتروی، فراهم می‌سازد. نظیر دولت، سیستم عامل هیچ عمل مفیدی به خودی خود اجرا نمی‌کند. آن، تنها محیطی^۲ که درون آن سایر برنامه‌ها بتوانند کار مفید انجام دهند، ایجاد می‌نماید.

می‌توانیم به سیستم عامل، به مثابه یک تخصیص دهنده منابع^۳، بنگریم. یک سیستم کامپیوتروی، منابع بسیاری (نرم‌افزار، سخت‌افزار) دارد که ممکن است در حل یک مسئله لازم باشند: زمان CPU، فضای حافظه، فضای ذخیره فایل، وسایل I/O، وغیره. سیستم عامل همانند مدیر منابع عمل می‌کند و آنها را بر حسب نیاز به برنامه‌های شخصی تخصیص می‌دهد. از آنجایی که تقاضاهای منابع بسیار زیادی - احتمالاً متقاض، می‌توانند وجود داشته باشند، سیستم عامل بایستی تصمیم بگیرد که به کدام تقاضا، منابع تخصیص داده شوند تا سیستم کامپیوتروی عادلانه^۴ و سودمند عمل نماید.

یک دید تسبیت متفاوت از سیستم عامل، بر روی نیاز کنترل وسایل I/O گوناگون و برنامه‌های کاربر، شمرکر می‌شود. سیستم عامل، برنامه کنترل است. یک برنامه کنترل، اجرای برنامه‌های کاربر را کنترل می‌کند تا از خطاهای استفاده غیر صحیح از کامپیوتر جلوگیری نماید. آن، به ویژه، به عملکرد و کنترل وسایل I/O مربوط می‌شود. عموماً، به‌حال، تعریف کاملاً کافی از یک سیستم عامل وجود ندارد. سیستم‌های عامل وجود دارند زیرا آنها بک روش مستدل، برای حل مسئله ایجاد یک سیستم محاسباتی قابل استفاده، می‌باشند. هدف اساسی سیستم‌های کامپیوتروی، اجرای برنامه‌های کاربر و نیز سهولت حل مسائل کاربر، می‌باشد. با سمت‌گیری این هدف، سخت‌افزار کامپیوتر ساخته شده است. از آنجایی که سخت‌افزار تنها، به ویژه سهل قابل استفاده نیست، برنامه‌های کاربردی توسعه یافته‌اند. این برنامه‌های متعدد، اعمال مشخص مشترکی را لازم دارند، از قبیل آنها بی که وسایل I/O را کنترل می‌نمایند. کار مشترک کنترل و تخصیص منابع، توأم، در یک نرم‌افزار، یعنی سیستم عامل، تلفیق شده است.

هیچ تعریف جامع و قابل قبولی از اینکه چه چیز بخشی از سیستم عامل است و چه چیز، نیست، هم وجود نماید.

1. government
4. fairly

2. environment

3. resource allocator

ندارد. یک دیدگاه ساده آن است که هر چیزی که یک فروشنده، در سفارش شما به عنوان «سیستم عامل»، ارسال می‌کند، می‌بایست در نظر گرفته شود. نیازمندیهای حافظه و ویژگیهای آن، به‌هرحال مابین سیستم‌ها، بسیار وسیع، تغییر می‌کنند. بعضی‌ها کمتر از یک مگابایت فضای خواهند و حتی از داشتن یک ویرایشگر تمام صفحه، کمبود دارند. در حالیکه بقیه صدها مگابایت فضای گیرند و شامل چک‌کننده‌های تلفظ و تمام «سیستم‌های پنجره‌ای» هستند. یک تعریف معمول‌تر آن است که در برنامه‌ای است که در تمامی اوقات بر روی سیستم کامپیوتر در حال اجرا است (معمول‌اً، هسته^۱ نامیده می‌شود) و تمام بقیه، برنامه‌های کاربردی هستند. این تعریف اخیر، معمول‌تر است و تعریفی است که ما عموماً، دنبال می‌کنیم.

~~ساده‌تر است که سیستم‌های عامل توسط کاری که انجام می‌دهند و نه آنچه که هستند، تعریف شوند. هدف اولیه سیستم عامل ایجاد سهولت برای کاربر، است. سیستم‌های عامل وجود دارند زیرا، مفروض است که با آنها کار کردن، آسان‌تر از، بدون آنها کار کردن، است. این دید، به ویژه، در نگرش شما به سیستم عامل برای کامپیوترهای شخصی کوچک، شفاف است.~~

هدف ثانویه، بازدهی سیستم کامپیوتری است. این هدف، به ویژه برای سیستم‌های چند کاربره مشترک و بزرگ، مهم است. این سیستم‌ها، به طور نمونه گران هستند. لذا مطلوب است که آنها را به قدر کافی، بهره‌ور سازیم. این دو هدف - سهولت^۲ و بهره‌وری^۳ - گاهی نقیض هم هستند. در گذشته، مفروضات بهره‌وری، غالباً مهم‌تر از سهولت، بوده است. بنابراین، یشتراوری سیستم‌های عامل، بر روی استفاده بهینه از منابع محاسباتی، تمرکز دارد.

گذشته، بررسی کنیم. با دنبال کردن این تکامل، می‌توانیم عناصر معمول سیستم‌های عامل را شناسایی نماییم، و بینیم چگونه و چرا این سیستم‌ها تکامل یافته‌اند.

سیستم‌های عامل و معماری کامپیوتر، اثر وسیعی بر یکدیگر داشته‌اند. به منظور سهولت کار با سخت‌افزار، در طراحی سخت‌افزار، می‌تواند آنها را ساده‌تر سازد. در این مرور تاریخی کوتاه، نتیجه گیری نمایید که چگونه مسائل سیستم‌های عامل، منجر به معرفی ویژگیهای سخت‌افزاری جدید گردیده است.

3. efficiency

~ convenience

۲.۱. سیستم‌های ساده دسته‌ای

۱.۲. Simple Batch Systems

کامپیوترهای اولیه، اکثراً (بطور فیزیکی) ماشین‌های بزرگی بودند که از طریق یک کنسول^۱ کار می‌کردند. ابزارهای ورودی معمول، کارت‌خوان‌ها و گردااندنه‌های نوار بودند. ابزارهای خروجی معمول، چاپگرهای خطی، گردااندنه‌های نوار، و ماشین‌های منگنه کارت، بودند. کاربران این چنین سیستم‌هایی، به طور مستقیم با سیستم‌های کامپیوتری، محاوره^۲ نداشتند. بلکه، یک کاربر، کاری^۳ را تهیه می‌کرد - که شامل یک برنامه، داده‌ها و تعدادی اطلاعات کنترلی درباره طبیعت کار (کارت‌های کنترلی) بود - و آن را به اپراتور کامپیوتر تحويل می‌داد. کار، معمولاً به فرم کارت‌های منگنه شده بود. در یک زمان دیرتر (شاید دقایق، ساعت‌ها، یا روزها)، خروجی ظاهر می‌شد. خروجی شامل نتایج برنامه و نیز یک رونوشت^۴ حافظه و رجیسترها در صورت خطای برنامه، بود.

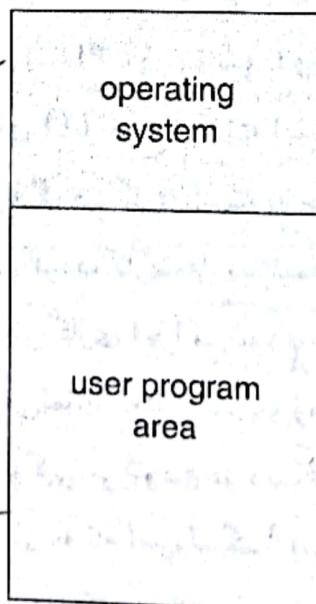
۷.۱ (سیستم عامل در این کامپیوترهای اولیه، نسبتاً ساده بود. وظیفه اصلی آن انتقال کنترل اتوماتیک از یک کار به دیگری بود. سیستم عامل همواره، مقیم^۵ در حافظه بود (شکل ۲-۱)).

حاصلت اساسی سیستم‌های کامپیوتری دسته اول:

خریدار
چالنجرها
کارت صوان حامر
گردااندنه‌ها نوار

کاربران سیستم‌ها را مسروک محاوره نمودند

→ اپراتور → کارت صوان نتریل
← خروجی‌ها را می‌نمودند
کاربران



شکل ۲-۱: طرح حافظه برای سیستم دسته‌ای ساده

- 1. console
- 2. interaction
- 3. job
- 4. dump
- 5. resident

مستقر

جهت تسريع پردازش، کارهای با نیازهای مشابه، به هم دیگر دسته^۱ می‌شدند، و توسط کامپیوتر به عنوان یک گروه، اجرا می‌گردیدند. لذا، برنامه‌نویس‌ها، برنامه‌هایشان را نزد اپراتور جا می‌گذاشتند. اپراتور برنامه‌ها را برسی نیازهای مشابه‌شان، منظم دسته‌بندی می‌کرد و وقتی که کامپیوتر در دسترس بود، هر دسته را اجرا می‌نمود. خروجی یعنی کار، به برنامه‌نویس مربوطه، ارسال می‌شد.

یک سیستم عامل دسته‌ای، به طور طبیعی، یک جریان از کارهای جداگانه را می‌خواند (مثلاً از یک کارن خوان) هر یک با کارت‌های کنترل خودش که آن‌چه را که کار انجام می‌داد، از پیش تعریف می‌نمود. وقتی کار کامل می‌شد، خروجی اش مثلاً بر روی چاپگر خطی، به چاپ می‌رسید. یک ویژگی تعریف شده از سیستم دسته‌ای، کمپود محاوره^۲ مابین کاربر و کار، در اثناء اجرای کار، است. کار آماده شده و تحويل داده می‌شود، و بعد از زمانی دیرتر، خروجی ظاهر می‌شود. تأخیر زمانی مابین تحويل کار و تکمیل کار (به نام زمان برگشت^۳) می‌تواند ناشی از زمان محاسبه مورد نیاز یا از تأخیر زمانی سیستم عامل قبل از شروع پردازش کار، نتیجه شود.

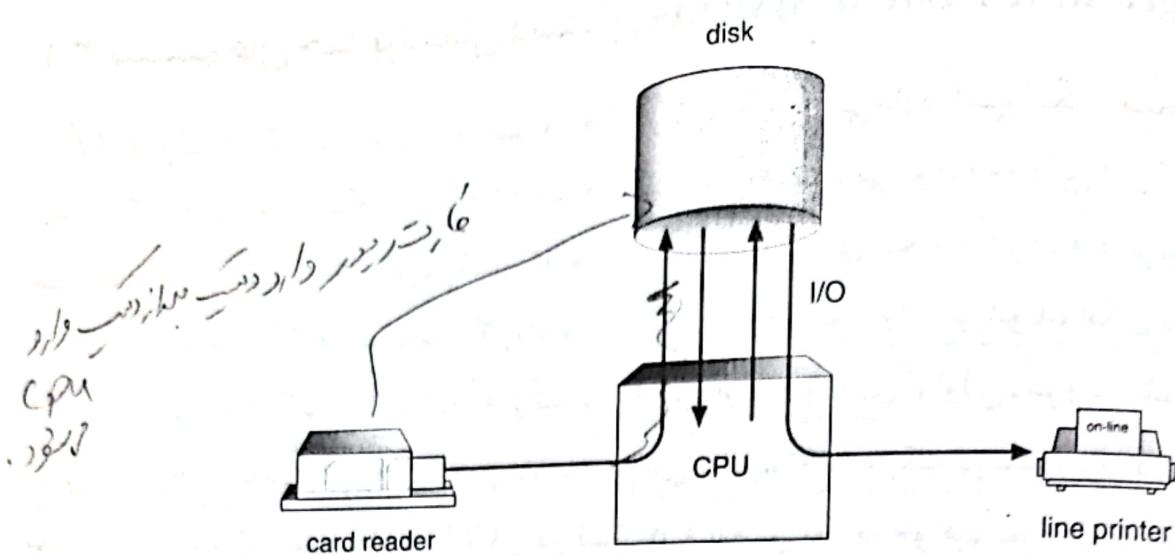
در این محیط اجرایی، CPU غالباً یکار^۴ است. این بیهوده ماندن بدین دلیل رخ می‌دهد که سرعت ابزار مکانیکی I/O، کندتر از ابزار الکترونیکی CPU است. حتی یک CPU گند در رنج میکروثانیه کار می‌کند و هزاران دستور العمل در ثانیه اجرا می‌شوند. از سوی دیگر یک کارت خوان سریع، می‌تواند ۱۲۰۰ کارت را در یک دقیقه بخواند (۱۷ کارت در ثانیه). بنابراین، سرعت عمل CPU در مقایسه با وسائل جانبی می‌تواند بسیار بیشتر باشد. در طی زمان، البته، پیشرفت تکنولوژی منجر به ابزارهای I/O سریع‌تر گشته است.

معرفی تکنولوژی دیسک، در این زمینه کمک کرده است (به جای آن که کارت‌ها از دستگاه کارت خوان مستقیماً وارد حافظه گردند و سپس کار، پردازش گردد، کارت‌ها، مستقیماً وارد دیسک می‌شوند. مکان تصویر کارت‌ها در جدولی توسط سیستم عامل ثبت می‌شود. وقتی کاری اجرا می‌شود، سیستم عامل تقاضاهای آن را برای ورودی از دستگاه کارت خوان، از روی دیسک، ارضاء می‌نماید. به طرز مشابه، زمانی که کار برای خروج یک خط، چاپگر را احضار می‌کند، خط خروجی در یک بافر سیستم کپی می‌شود و در دیسک نوشته می‌شود و وقتی که کار تکمیل گردید، خروجی، عملاً، چاپ می‌شود. این نحوه پردازش به نام اسپولینگ^۵ (شکل ۳-۱) نامیده می‌شود؛ نام برگرفته از مخفف زیر نوشته است. در این روش، در واقع، دیسک به مانند یک بافر وسیع، برای خواندن هر چه بیشتر از دستگاه ورودی و نگهداری فایل‌های خروجی انگاشته می‌شود، تا زمانی که ادوات خروجی قادر به قبول خروجی‌ها، شوند.

- ۱. batch
- ۴. idle

- 2. interaction
- 5. spooling: simultaneous peripheral operation on-line
- 3. turnaround time

شکل ۳-۱- تفاوت ۷۰۰۰ر، راز ۶ رتران از درایور برای کردن



شکل ۱-۳: اسپولینگ

اسپولینگ، همین طور، برای پردازش داده در سایت‌های دور^۱ به کار می‌رود. CPU، داده را از مسیرهای ارتباطی به یک چاپگر دور ارسال می‌کند (یا کل کار را از یک دستگاه کارت‌خوان دور قبول می‌کند). پردازش دور^۲ در میزان سرعت خود، بدون مداخله CPU انجام می‌گیرد. CPU فقط باستی توجه نماید که چه موقع پردازش تکمیل باشه است، تا دستهٔ بعدی داده‌ها را جمع آوری نماید.

اسپولینگ، عمل I/O یک کار را با عمل محاسباتی کار دیگر روی هم^۳ می‌اندازد. حتی در یک سیستم ساده، اسپولر^۴ می‌تواند ورودی یک کار را از دستگاه ورودی بخواند در حالی که، در حال چاپ خروجی کار دیگری است. در این اثناء، حتی، کار دیگری (یا کارهای دیگری) می‌توانند در حال اجرا باشند، ورودی‌هایشان را از «کارت‌های» روی دیسک بخوانند و خروجی‌هایشان را بر روی دیسک «چاپ» نمایند.

اسپولینگ، اثر مستقیم بر بهبود، کارآیی^۵ سیستم دارد. در ازای هزینهٔ مقداری فضای ذخیره و تعداد اندکی جدول، محاسبات یک کار با اعمال ورودی/خروجی کار دیگر رویهم می‌افتد. لذا هم CPU و هم ابزارهای I/O به

میزان بالاتری کار مفید انجام می‌دهند.

- 1. remote site
- 4. spooler

- 2. remote processing
- 5. performance

- 3. overlap

کاری

۳.۱. سیستم‌های چند برنامه‌ای دسته‌ای

(۱) اسپولینگ یک ساختمان داده مهم به نام انبارکار^۱، فراهم می‌سازد. اسپولینگ، منجر به وجود کارهای متعددی که قبل بر روی دیسک خوانده شده‌اند و منتظر اجرا^۲ هستند، می‌شود. وجود این انبار کار بر روی دیسک، به سیستم عامل اجازه می‌دهد تا تصمیم بگیرد کدام کار برای اجرای بعدی انتخاب گردد و ضریب استفاده^۳ CPU را افزایش می‌دهد. وقتی کارها، مستقیماً از کارت خوان یا گردانده‌های نوار وارد می‌شوند، امکان ندارد که اجرای آنها به ترتیب متفاوتی، انجام گیرد. کارها باید به ترتیب، بر اساس اولین-ورودی^۴، اولین-سرویس شده، صورت گیرد. اما زمانی که کارهای متعددی بر روی یک رسانه با دسترسی مستقیم^۵، مانند یک دیسک، قرار دارند، زمان‌بندی کار، ممکن می‌گردد. ما زمان‌بندی کار و CPU را در فصل ۵ به تفصیل مطالعه خواهیم نمود. در اینجا، فقط، تعداد کمی از جنبه‌های مهم، تحت پوشش قرار گرفته است.

مهمترین وجه زمان‌بندی کار، توانایی چند برنامگی است. عمل خارج-خط^۶ و اسپولینگ، برای رویهم انداختن I/O، محدودیت‌های خود را دارند. یک کاربرد منفرد، نمی‌تواند، هم CPU و هم ابزارهای I/O را در تمامی اوقات مشغول نگهداشد. چند برنامگی، با این سازماندهی که CPU همواره یک کار برای اجرا دارد، سبب افزایش درصد استفاده CPU می‌گردد.

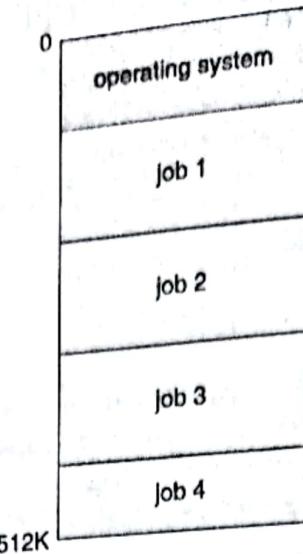
(۲) ایده بشرح زیر است. سیستم عامل کارهای متعددی را، در یک زمان، در حافظه نگه می‌دارد (شکل ۴-۱). این مجموعه کارها، یک زیرمجموعه از کارهای نگهداشتی شده در اباتار کارهاست (زیرا تعداد کارهای نگهداشته شده در حافظه اصلی خیلی کمتر از تعداد کارهای انبار کار، بر روی دیسک است). سیستم عامل یک کار را برگرفته و آن را اجرا می‌کند. بالاخره، کار ممکن است برای انتظار یک وظیفه، مانند سوارشدن^۷ یک نوار، یا تکمیل یک عمل I/O، مجبور گردد. در یک سیستم تک برنامگی، CPU، بیهوده انتظار خواهد کشید. در یک سیستم چند برنامگی، سیستم عامل به سادگی به کار دیگری سوئیچ نموده و آن را اجرامی کند. وقتی که آن کار هم، نیاز به انتظار داشت، CPU به کار دیگری سوئیچ می‌کند، و الی آخر. بالاخره، کار اول انتظار را به اتمام می‌رساند و CPU را مجدداً پس می‌گیرد. آن جایی که همواره یک کار برای اجرا وجود دارد، CPU هرگز بیکار و بیهوده نمی‌ماند.

- 1. job pool
- 4. first-come first-served
- 7. off-line

- 2. run
- 5. direct-access
- 8. mounting a tape

- 3. utilization
- 6. job scheduling

مفهوم سیستم عامل ۱۱



شکل ۱-۴: طرح حافظه برای یک سیستم چند برنامگی

این ایده در سایر وضعیت‌های زندگی هم متدالو است. یک وکیل فقط یک مشتری، در یک زمان، ندارد. بلکه مشتری‌های متعددی همزمان در فرآیند سرویس دهی و خدمات‌گیری می‌توانند وجود داشته باشند. در حالیکه یک مورد متظر آزمایش است یا منتظر مدارکش تا تایپ شوند، وکیل می‌تواند بر روی مورد دیگری کار کند. اگر به تعداد کافی مشتری داشته باشد، هرگز وقش بیهوده تلف نمی‌شود. (وکلای یکار، تعایل به سیاستمدار شدن پیدا می‌کنند، لذا ارزش اجتماعی مشخصی در مشغول نگهداشتن وکلا، وجود دارد.)

(مالتی پروگرامینگ)، اولین نمونه از زمانی است که سیستم عامل باستی به جای کاربران تصمیم بگیرد. سیستم‌های عامل چند برنامه‌ای، لذا، نسبتاً پیچیده^۱ هستند. کلیه کارهایی که وارد سیستم می‌شوند، در اینبار کار نگهداشته می‌شوند. این اینبار شامل کلیه پراسس‌هایی است که بر روی دیسک قرار دارند و منتظر تخصیص یافتن حافظه اصلی به آنها، می‌باشند اگر کارهای متعددی منتظر ورود به حافظه باشند و فضای کافی برای همگی در دسترس نباشد، در این صورت، سیستم عامل باید تعدادی را برگزیند. انجام این تصمیم‌گیری، زمانبندی^۲ کار نامیده می‌شود، که در فصل ۵ مورد بحث قرار خواهد گرفت. زمانی که، سیستم عامل کاری را از اینبار کار^۳، برگزید، آن را در حافظه بار می‌کند تا اجرا گردد. داشتن کارهای متعدد همزمان در حافظه، نیازمند داشتن یک فرم مدیریت حافظه است، که در فصل‌های ۸ و ۹ پوشش خواهد یافت. به علاوه، (اگر کارهای متعددی، همزمان آماده اجرا باشند، سیستم باستی از میان آنها یکی را

زمانبندی)

1. sophisticated

2. job scheduling

3. job pool

زمانبندی کار، ورود حافظه

زمانبندی CP4 کار، مسوده همزمان، اجرا

برگزیند. انجام این تصمیم‌گیری، زمانبندی CPU^۱ نامیده می‌شود و در فصل ۵ مورد بحث قرار خواهد گرفت. بالاخره، کارهای متعدد در حال اجرای هموزنده^۲، ملزم می‌دارند که توانایی شان در اثر کردن به یکدیگر، در کلیه فازهای سیستم عامل، شامل زمانبندی پراسس، ذخیره دیسک، و مدیریت حافظه، محدود گردد. این مفروضات در طبق این متن، مورد بحث قرار خواهند گرفت.

1.4. Time-Sharing Systems

۱.۴.۱. سیستم‌های اشتراک - زمانی

سیستم‌های چند برنامه‌ای دسته‌ای، محیطی فراهم می‌سازند که منابع مختلف سیستم (مثلًا CPU، حافظه، دستگاه‌های جانبی) بطور سودمند مورد استفاده قرار گیرند. در سیستم‌های دسته‌ای مشکلاتی از زاویه دید کاربر، به‌هرحال مشهود است. از آن جایی که، کاربر نمی‌تواند در هین اجرای کار، با آن محاوره داشته باشد، بایستی کارت‌های کنترلی را برای انجام تمامی نتایج، تنظیم نماید. در یک کار چند-گام، گام‌های پشت سر هم، ممکن است به نتایج گام‌های ماقبل، وابسته باشند. اجرای یک برنامه، به عنوان مثال، تابع کامپایل موفقیت‌آمیز، است. تعریف کامل آنچه که باید در کلیه موارد انجام گیرد، مشکل است.

مشکل دیگر، آن است که برنامه‌ها باید به صورت ایستا، از رونوشت لحظه‌ای^۳، خطایابی^۴ شوند. برنامه‌نویس نمی‌تواند برنامه در حال اجرا را تغییر دهد و رفتار آن را مطالعه نماید. مدت زمان برگشت^۵ طولانی، آزمایش را منع می‌کند.

(اشتراک - زمانی، یا چند وظیفه^۶ کی^۷، تعمیم منطقی چند برنامگی است. کارهای متعددی توسط سوئیچ کردن CPU مابین آنها، اجرا می‌شوند، اما، سوئیچ کردن‌ها به قدری، مکرر، رخ می‌دهند، که کاربر می‌تواند با برنامه در هین اجزا، محاوره داشته باشد.)

یک سیستم کامپیوتری محاوره‌ای^۸ یا دست-بر^۹، ارتباط بر-خط^{۱۰}، مابین کاربر و سیستم برقرار می‌سازد. کاربر مستقیماً، دستور العمل‌هایی را به سیستم عامل یا به برنامه، می‌دهد، و پاسخ آنی^{۱۱}، دریافت می‌نماید. یک صفحه کلید، برای فراهم نمودن ورودی و یک صفحه نمایش (مانند لامپ اشعه کاتدیک^{۱۱} (CRT)، یا مانیتور) برای فراهم نقل‌کردن مراحل بیشتر.

- 1. CPU scheduling
- 2. concurrent
- 4. debug
- 5. turnaround time
- 7. interactive
- 8. hands-on
- 10. immediate response
- 11. cathode-ray tube

- 3. snapshot dumps
- 6. multitasking
- 9. on-line

نمودن خروجی به کار می‌روند. وقتی که سیستم عامل، اجرای یک فرمان^۱، را به اتمام رساند، «حکم کنترلی» دیگری را، نه از کارت خوان، بلکه از صفحه کلید کاربر، جستجو می‌نماید. کاربر، فرمانی را وارد می‌کند و منتظر پاسخ می‌ماند و در مورد فرمان بعدی تصمیم‌گیری می‌کند که بستگی به نتیجه فرمان قبلی دارد. کاربر، به آسانی آزمایش می‌کند و می‌تواند فوری نتایج را مشاهده نماید. اکثر سیستم‌ها، دارای ویرایشگر متن محاوره‌ای برای وارد کردن برنامه و نیز دیگر^۲‌های محاوره‌ای جهت دستیاری عمل خطایابی برنامه‌ها، می‌باشند.

اگر کاربران قادر به دسترسی کد و داده به سهولت هستند، بایستی یک سیستم فایل بر-خط، در دسترس باشد. فایل، یک مجموعه از اطلاعات مربوط به هم، تعریف شده توسط ایجاد کننده خود، می‌باشد. معمولاً، فایل در برگیرنده برنامه‌ها (هم فرم منبع^۳ و هم فرم مقصد^۴) و داده می‌باشد. فایل‌های داده‌ای ممکن است عددی، الفبایی، یا الفبا عددی، باشند. فایل‌ها ممکن است بدون قالب^۵، مانند فایلهای متی، و یا قالب شده^۶، باشند. معمولاً، فایل یک دنباله^۷ از بیت‌ها، بایت‌ها، خطوط، یا رکوردهایی که معانی آنها توسط ایجاد کننده یا کاربر فایل، تعریف شده‌اند، می‌باشد. سیستم عامل، مفهوم انتزاعی^۸ فایل را، با مدیریت رسانه‌های ذخیره^۹ انبوه، مانند نوارها و دیسک‌ها، پیاده‌سازی می‌نماید. فایل‌ها، بطور طبیعی، به فرم کلاسترها منطقی^{۱۰}، یا دایرکتوری‌ها^{۱۱}، سازمان دهی می‌شوند که سبب سهولت یافتن و دسترسی آنها می‌شود. از آن جایی که، کاربران متعددی، به فایل‌ها دسترسی دارند، مطلوبست که کنترل شود فایل‌ها توسط چه کسی و به چه طریقی مورد دسترسی واقع شده‌اند.

سیستم‌های دسته‌ای، برای اجرای برنامه‌های بزرگ که نیاز محاوره‌ای کمی دارند، مناسب می‌باشند. کاربر می‌تواند کار را تحویل داده و در زمان دیرتری برای کسب نتیجه خروجی، بازگردد؛ لزومی ندارد که کاربر منتظر اجرا باشد. کارهای محاوره‌ای، تمایل به تجزیه به اعمال کوتاه بسیاری، دارند و در عین حال نتیجه دستور بعدی ممکن است غیر قابل پیش‌بینی باشد. کاربر فرمان را وارد می‌کند و منتظر پاسخ می‌ماند. بدین مناسبت، زمان پاسخ^{۱۲} بایستی کوتاه باشد - حداقل در مقیاس ثانیه‌ها. یک سیستم محاوره‌ای، وقتی که زمان پاسخ کوتاه نیاز است، به کار می‌رود.

سیستم‌های اولیه، با کاربر منفرد، سیستم‌های محاوره‌ای بودند. یعنی، کل سیستم در خدمت آنی برنامه‌نویس / اپراتور، قرار داشت. این وضعیت به برنامه‌نویس بیشترین قابلیت انعطاف و آزادی را در تست و توسعه برنامه، می‌داد. اما، همانگونه که مشاهده شد، این تشکیلات منجر به اتلاف زمان CPU در موقعی که برنامه منتظر عملی

- 1. command
- 4. object
- 7. sequence
- 10. cluster

- 2. debugger
- 5. free-format
- 8. abstract
- 11. directory

- 3. source
- 6. formatted
- 9. mass-storage
- 12. response time

از جانب کاربر یا برنامه نویس می‌باشد، می‌گردید. به دلیل هزینه بالای سیستم‌های قدیمی، زمان بیهوذه CPU مطلوب نبود. سیستم‌های عامل دسته‌ای برای اجتناب از این مسئله، توسعه یافتد. سیستم‌های دسته‌ای، بهره‌وری را برای صاحبان سیستم‌های کامپیوتری، بهبود بخشیدند.

سیستم‌های اشتراک - زمانی، برای فراهم نمودن استفاده محاوره‌ای از سیستم کامپیوتری در ازای هزینه مقبولی^۱ توسعه یافتد. هر کاربر، حداقل یک برنامه در حافظه دارد. برنامه‌ای که در حافظه بار شده است و در حال اجراست، معمولاً به نام پردازش^۲ یاد می‌شود. وقتی پردازشی اجرا می‌شود، برای مدت کوتاهی، قبل از آن که تمام شود یا نیاز به عمل I/O داشته باشد، اجرا می‌گردد. I/O ممکن است محاوره‌ای باشد یعنی خروجی در مانیتور نمایش داده شود و ورودی از صفحه کلید کاربر، گرفته شود. چون ورودی / خروجی محاوره‌ای به طور نمونه، با سرعت انسان انجام می‌گیرد، زمان بیشتری برای تکمیل ممکن است لازم باشد. مثلاً ورودی، محدود به سرعت تایپ کاربر می‌شود؛ ۵ کرکتر در یک ثانیه، برای یک انسان نسبتاً سریع است، اما به طور غیرقابل انکار، برای کامپیوتر، کند، است. به جای آنکه CPU در این فاصله محاوره ورودی، بیهوذه منتظر بماند، سیستم عامل سریعاً به برنامه کاربر دیگری سوئیچ می‌نماید.

۱۴۹ (سیستم عامل اشتراک - زمانی، اجازه می‌دهد کاربران بسیاری به طور همزمان سیستم کامپیوتری را به کار بزنند: چون، هر عمل یا فرمان در سیستم اشتراک - زمانی تمایل به زمان کوتاه دارد، تنها زمان کمی از CPU برای هر کاربر، نیاز است. اگرچه سیستم، سریعاً از یک کاربر به دیگری، سوئیچ می‌نماید، هر کاربری تصور می‌کند که کامپیوتر خودش را دارد، در حالیکه عمالاً یک کامپیوتر مابین کاربران زیادی، به اشتراک گذاشته شده است.)

ایده سیستم‌های اشتراک - زمانی، پیش از سال ۱۹۶۰، پدید آمد، اما چون این سیستم‌ها مشکل و گران ساخته می‌شدند، قبل از اوایل سالهای ۱۹۷۰، معمول نبودند. همین گونه که رواج اشتراک - زمانی، رشد نمود، محققین سعی در ادغام سیستم‌های دسته‌ای و اشتراک - زمانی نمودند. بسیاری از سیستم‌هایی که صرفاً سیستم‌های دسته‌ای طراحی شده بودند، طوری تغییر یافتد تا زیر سیستم اشتراک - زمانی ایجاد نمایند. به عنوان مثال، IBM's OS/360، یک سیستم دسته‌ای، تغییر نمود تا گزینه اشتراک - زمانی (TSO)^۳ را حمایت نماید. همزمان، سیستم‌های اشتراک - زمانی، یک اگرچه طراحی و کاربرد اولیه آنها در جهت یکی از انواع ذکر شده، می‌باشد.

سیستم‌های عامل اشتراک - زمانی، حتی پیچیده‌تر از سیستم‌های عامل چند برنامگی هستند. همانند، سیستم‌های چند برنامگی، بایستی کارهای متعددی همزمان در حافظه نگه داشته شوند، که نیازمند نوعی مدیریت حافظه و محافظت

3. time-sharing option

1. reasonable cost

2. process

(فصل ۸) می باشد. اگرچه زمان پاسخ مقبولی می تواند به دست آید، کارها ممکن است مجبور به جایه جایی، به واز، حافظه اصلی و دیسک گردند که در این حالت، دیسک به عنوان ذخیره پشتیبان^۱ برای حافظه اصلی عمل می کند. روش معمول انجام این امر، حافظه مجازی^۲ نامیده می شود و تکنیکی است که اجازه اجرای یک کار را، که ممکن است تماماً در حافظه قرار نداشته باشد، میسر می سازد (فصل ۹). منفعت اصلی قابل مشهود این طرح، آن است که برنامه ها، می توانند بزرگتر از حافظه فیزیکی باشند. به علاوه، این تکنیک، حافظه اصلی را به یک بردار یکنواخت بزرگ معنی می کند و حافظه منطقی از دید کاربر را از حافظه فیزیکی جدا می سازد. این تشکیلات، کاربر را از محدودیت ذخیره حافظه، آزاد می سازد. سیستم های اشتراک - زمانی بایستی یک سیستم فایل بر - خط، فراهم سازند. (فصل ۱۰ و ۱۱). سیستم فایل بر روی مجموعه ای از دیسک ها واقع است، لذا، مدیریت دیسک نیز بایستی فراهم شود (فصل ۱۲). همین طور، سیستم های اشتراک - زمانی، مکانیزمی برای اجرای همروند ارائه می دهند که نیازمند طرح های زمان بندی CPU پیچیده، می باشد (فصل ۵). برای تضمین اجرای منظم، سیستم بایستی مکانیزمی برای سنکرون کردن و ارتباط پردازش ها ارائه دهد، و بایستی اطمینان دهد که کارها در یک بن بست، به انتظار دائمی یکدیگر، گیر نمی کنند (فصل ۷).

چند برنامگی و اشتراک - زمانی تم اصلی سیستم های عامل مدرن و نیز تم مرکزی متن این کتاب می باشد.

1.5. Personal-Computer Systems

۱.۵. سیستم های کامپیوتر شخصی

همین گونه که هزینه سخت افزار کاهش می یابد، یک بار دیگر، وجود یک سیستم کامپیوتری اختصاصی برای یک کاربر منفرد، ملموس می شود. این نوع سیستم های کامپیوتری، معمولاً، به نام کامپیوترهای شخصی (PC) رجوع می شوند. وسایل I/O، یقیناً تغییر کرده اند. صفحات^۳ سوئیچ ها و کارت خوان ها با ماشین های تله تایپ - گونه صفحه کلید و موس ها، جایگزین شده اند. چاپگرهای حجمی و منگنه های کارت با صفحه نمایش و چاپگرهای کوچک سریع، جایگزین گشته اند.

سیستم های کامپیوتری شخصی در سال ۱۹۷۰ ظاهر گردیدند. آنها ریز کامپیوترهای هستند که بسیار کوچکتر و ارزان تر از سیستم های بزرگ^۴ می باشند. در دهه اول، CPU در PC ها قادر ویژگی های لازم برای حافظت سیستم عامل از برنامه های کاربر، بود. بنابراین، سیستم های عامل PC ها، نه چند کاربره^۵ و نه چند وظیفه ای

mainframe

- | | | |
|--------------------|-------------------|------------------|
| 1. back-up storage | 2. virtual memory | 3. switch panels |
| 4. mainframe | 5. multi-user | |

بودند. به هر حال، اهداف این سیستم‌های عامل، در طول زمان، تغییر نموده است. به جای ماکریمم کردن در صد استفاده CPU و وسایل جانبی، سیستم به سمت راحتی کاربر و مسئولیت‌پذیری او، شکل گرفت. این سیستم‌ها، شامل PC‌هایی که ویندوز مایکروسافت را اجرا می‌کنند و نیز سیستم‌های اپل مکینتاش می‌باشند. ویندوز‌های متعدد مایکروسافت جایگزین سیستم عامل MS-DOS مایکروسافت گشته‌اند و شرکت IBM، MS-DOS را به سیستم عامل چند وظیفه‌ای OS/2 ارتقاء داده است. سیستم عامل اپل مکینتاش به سخت‌افزار پیشرفته‌تری اعمال شده است و هم اکنون ویژگی‌های جدید مانند حافظه مجازی را، دربرمی‌گیرد.

سیستم‌های عامل این کامپیوترها، از توسعه سیستم‌های عامل کامپیوتروهای بزرگ مبنی فریم^۱ به طرق متعدد، بهره جسته‌اند. ریزکامپیوترها، بلا فاصله قابلیت سازگاری با تکنولوژی پیشرفته برای سیستم‌های عامل بزرگ‌تر را، داشتند. از سوی دیگر، هزینه سخت‌افزار برای ریزکامپیوترها، به قدر کافی پایین آمد، به طوری که هر شخصی استفاده تنها از کامپیوتر می‌کند و در صداستفاده CPU دیگر فرض اصلی نیست. لذا، بعضی تصمیم‌گیری‌های طراحی در مورد سیستم‌های عامل کامپیوترهای بزرگ، ممکن است برای سیستم‌های کوچک‌تر، مناسب نباشند. به عنوان مثال، حفاظت فایل در کامپیوترهای شخصی ضروری به نظر نمی‌رسد.

بعضی افراد، مجادله کرده‌اند که پیشرفت ریزکامپیوترهای ارزان و حافظه ارزان، سیستم‌های عامل (و لذا دروس مربوط به آنها) را از دور خارج^۲ کرده‌اند. ما اعتقاد نداریم که این پیشگویی صحیح است. بلکه، کاهش قیمت سخت‌افزار، موجب می‌شود که مفاهیم نسبتاً پیچیده سیستم‌های عامل (نظیر حافظه مجازی و اشتراک-زمانی) در حتی تعداد بیشتری از سیستم‌ها، پیاده‌سازی شوند. لذا، کاهش هزینه سخت‌افزار کامپیوتر، نظیر ریزکامپیوترها، نیاز به درک مفاهیم سیستم‌های عامل را افزایش خواهد داد.

۱. ع (به عنوان مثال، اگرچه حفاظت فایل ممکن است در سیستم‌های کامپیوتری شخصی، ضروری به نظر نرسد، اما این کامپیوترها غالباً به کامپیوترهای دیگری از طریق خطوط ارتباطی یا شبکه‌های محلی، اتصال می‌یابند. وقتی، کامپیوترهای دیگر و کاربران دیگر بتوانند به فایل‌های کامپیوتر شخصی مزبور، دسترسی داشته باشند، حفاظت فایل، مجدداً یک ویژگی لازم سیستم عامل می‌گردد. کمبود چنین حفاظتی، سبب شده است که برنامه بدخواهی، به سادگی داده‌های سیستمی نظیر MS-DOS و اپل مکینتاش را، خراب نماید. این برنامه‌ها، ممکن است، خود-تکرار^۳ باشند. و به سرعت از طریق مکانیزم‌های ویروس^۴ یا کرم^۵ منتشر شده و تمامی شبکه کمپانی یا حتی شبکه‌های جهانی را، مختل سازند)

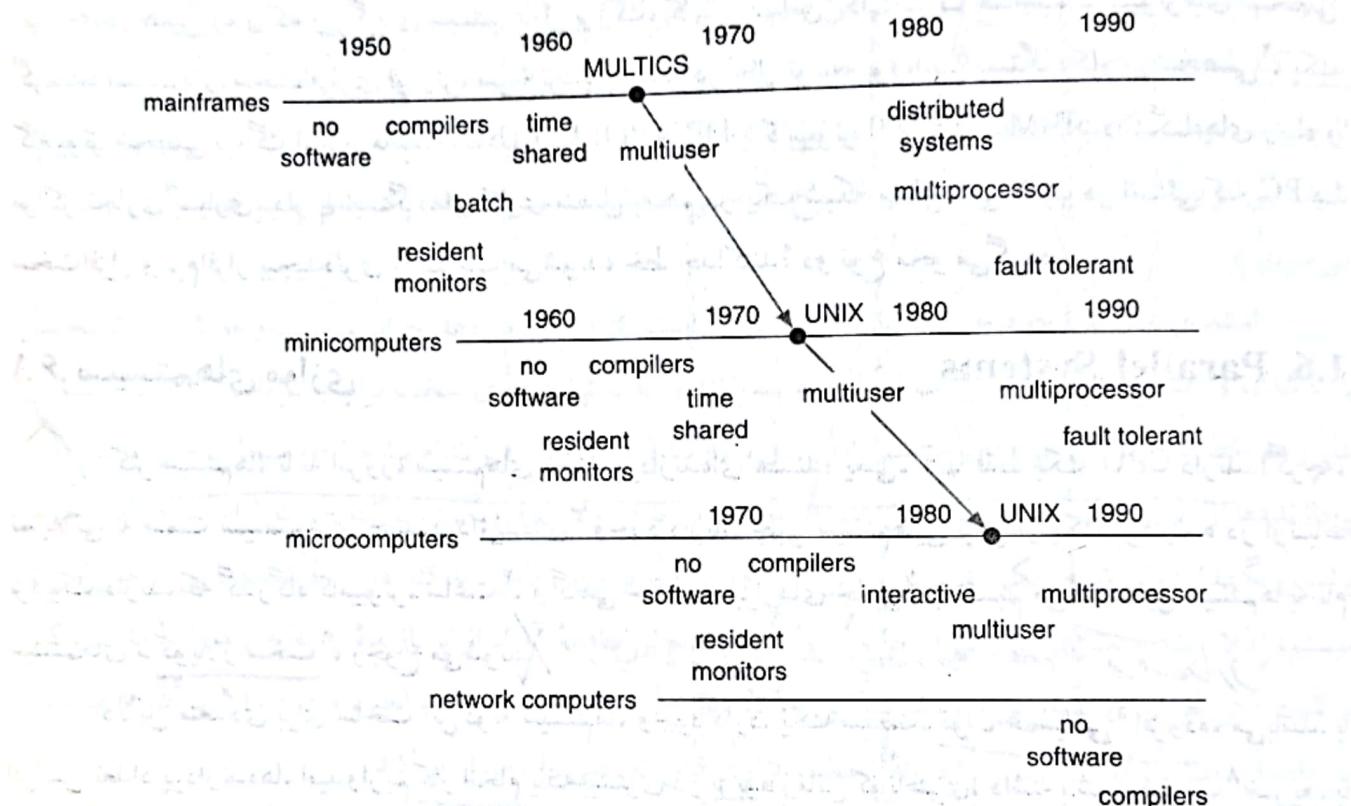
۲

- 1. main-frame
- 4. virus

- 2. obsolete
- 5. worm

- 3. self-replicating

در واقع، تست سیستم‌های عامل کامپیوترهای مین فریم و مایکروکامپیوترها، نشان می‌دهد که ویژگی‌هایی که در یک زمان، فقط در مین فریم‌ها در دسترس بود، توسط مایکروکامپیوترها نیز، اقتباس شده است. مفاهیم یکسانی، برای انواع متفاوت متعدد کامپیوترها؛ مین فریم‌ها، مینی کامپیوترها و مایکروکامپیوترها (شکل ۱-۵) مناسبند.



شکل ۱-۵: مهاجرت مفاهیم و ویژگی‌های سیستم عامل

یک مثال خوب از این حرکت، در سیستم عامل مالتیکس^۱ رخ داد. مالتیکس در سالهای ۱۹۶۵ تا ۱۹۷۰ در دانشگاه MIT، به عنوان یک یوتیلیتی^۲ محاسباتی، توسعه یافت. و بر روی کامپیوتر مین فریم پیچیده GE 645 اجرا شد. بسیاری از ایده‌هایی که برای مالتیکس شکل گرفته بود، متعاقباً در لابراتوارهای بل^۳ (یکی از شرکت‌کننده‌های اصلی پروژه مالتیکس) در طراحی یونیکس^۴ به کار گرفته شد. سیستم عامل یونیکس در حدود ۱۹۷۰ برای مینی

- 1. multics
- 2. utility
- 3. Bell Labs
- 4. UNIX

کامپیوتر PDP-II طراحی شد. در حدود ۱۹۸۰، ویژگیهای یونیکس، یک مینا برای سیستم‌های عامل یونیکس-گونه مایکرو کامپیوترها گشت، و در بسیاری از سیستم‌های عامل اخیر نظری مایکروسافت ویندوز IBM OS/2، NT و Macintosh Operating System شامل گردید. بنابراین، ویژگیهای توسعه یافته سیستم مین فریم بزرگ در طول زمان، به سوی مایکرو کامپیوترها، حرکت کرده است.

در همین زمان که ویژگیهای سیستم عامل بزرگ، کاهش مقیاس^۱ داده‌اند تا مناسب کامپیوترهای شخصی گردند، سیستم‌های سخت‌افزاری قوی‌تر، سریع‌تر، پیچیده‌تر در حال توسعه بوده‌اند. ایستگاه کاری شخصی^۲، یک کامپیوتر شخصی بزرگ است، مانند SUN، HP/APOLLO یا کامپیوتر IBM RS/6000. دانشگاه‌های زیاد و مراکز تجاری بسیاری، دارای ایستگاه‌های کاری متصل به هم در یک شبکه محلی، می‌باشند. در اثنائی که PC‌ها سخت‌افزار و نرم‌افزار پیچیده‌تری را صاحب می‌شوند، خط جداکننده دو نوع محو می‌گردد.

1.6. Parallel Systems

۱.۶. سیستم‌های موازی

اکثر سیستم‌ها، تا به امروز، سیستم‌های تک-پردازنده‌ای هستند؛ یعنی، آنها فقط یک CPU دارند. اگرچه، تمایلاتی به سمت سیستم‌های چند پردازنده‌ای^۳ وجود دارند. چنین سیستم‌هایی بیش از یک پردازنده در ارتباط نزدیک دارند، که گذرگاه کامپیوتر، ساعت^۴، و گاهی حافظه و ابزارهای جانبی^۵ را تقسیم می‌کنند. این سیستم‌ها به نام سیستم‌های با کوپل‌از سخت^۶، رجوع می‌شوند. ~~برای نفع پردازنده‌ها~~ پردازنده حافظه سریع‌کرده‌اند. دلایل متعددی برای ساخت این گونه سیستم‌ها، وجود دارد. یک منفعت، توان عملیاتی^۷ افزوده، می‌باشد. با افزایش تعداد پردازنده‌ها، امیدواریم کار انجام یافته بیشتری در پریود زمانی کوتاه‌تری، داشته باشیم. ضریب^۸ تسریع، با داشتن n پردازنده، n نیست، بلکه کمتر از n است. وقتی که پردازنده‌های متعددی بر روی یک وظیفه همکاری می‌کنند، بارس^۹ مشخصی جهت حفظ کار درست تمامی قسمت‌ها، رخ می‌دهد. این بارس به علاوه رقابت برای منابع تقسیم شده، انتظار سود از پردازنده‌های اضافی را، کاهش می‌دهد. به طرز مشابه، یک گروه از n برنامه‌نویس که بسیار نزدیک باهم کار می‌کنند، منجر به n برابر کار انجام یافته، نمی‌شود. ~~مالی پراسورها، می‌توانند، نیز، در صرفه‌جویی هزینه، در مقایسه با سیستم‌های منفرد چندگانه، مطرح شوند، زیرا پردازنده‌ها می‌توانند ادوات جانبی، کابینت‌ها، و منابع تغذیه را تقسیم نمایند. اگر برنامه‌های متعددی بر روی~~

- 1. scale down
- 4. clock
- 7. throughput

- 2. workstation
- 5. peripheral
- 8. speed-up ratio

- 3. multiprocessor
- 6. tightly-coupled systems
- 9. overhead

سیستم‌های متعددی برای سیستم تحقیلی سود را همراه با خود نمایند.

مجموعهٔ یکسانی از داده‌ها، عمل کنند، ارزان‌تر است که داده‌ها در یک دیسک ذخیره نمود و در تمام پردازنده‌ها آنها را به اشتراک گذاشت به جای آن که، سیستم‌های تک پردازنده متعددی داشت و دیسک‌های محلی و کپی‌های بیشماری از داده بر روی آنها، ذخیره نمود.

دلیل دیگر برای سیستم‌های مالتی پراسپور، آن است که آنها قابلیت اعتماد را افزایش می‌دهند. اگر اعمال به درستی مابین پردازنده‌ها توزیع شود، در این صورت خرابی^۱ یک پراسپور، سبب توقف سیستم نمی‌گردد، بلکه تنها سبب کند شدن آن خواهد شد، اگر ۱۰ پردازنده و یک خرابی داشته باشیم، به جای خراب شدن کل، تنها ۱۰ درصد از سرعت سیستم، کاسته خواهد شد، این توانایی به ادامه سرویس، متناسب با سطح بقای سخت افزار، تنزل مطبوع^۲ نامیده می‌شود. سیستم‌هایی که برای تنزل مطبوع طراحی می‌شوند، سیستم‌های تحمل پذیر خطای^۳ نیز، نامیده می‌شوند.

استمرار عمل با وجود خرابی، نیازمند مکانیزمی است که اجازه می‌دهد خرابی، جست‌وجو شده، تشخیص داده شده، و اصلاح شود (در صورت امکان). سیستم تندم^۴ با تکثیر هردوی سخت افزار و نرم افزار، ادامه عمل را علی‌رغم خطاهای اطمینان می‌دهد. سیستم دارای دو پردازنده همگون، هر کدام با حافظه محلی خود، می‌باشد. پردازنده‌ها از طریق گذرگاه به هم وصل هستند. یک پردازنده، اولیه^۵، و دیگری یدک پشتیبان^۶ است. دو کپی از هر پراسس نگهداری می‌شود؛ یکی بر روی ماشین اولیه و دیگری بر روی ماشین یدک. در نقاط تست ثابتی^۷، حين اجرای سیستم، اطلاعات حالت هر پراسس (شامل یک کپی از تصویر حافظه آن)، از پردازنده اولیه به پردازنده یدک، کپی می‌شود. اگر خرابی‌ای تشخیص داده شود، کپی یدک فعال می‌شود و از نقطه تست اخیر، مجدداً، از سرگرفته می‌شود. این راه حل، مسلماً، گران است زیرا دوبل^۸ سخت افزاری قابل توجهی را تحمیل می‌کند.

اما متدال ترین سیستم‌های چند پردازنده‌ای امروزی، مدل چند پردازنده‌ای مستقarn^۹ است که در آن هر پردازنده، کپی همسانی^{۱۰} از سیستم عامل را اجرا می‌کند و این کپی‌ها بر حسب نیاز، با یکدیگر ارتباط برقرار می‌کنند. بعضی سیستم‌ها مدل چند پردازنده‌ای نامتنازن^{۱۱}، را به کار می‌برند، که در آنها هر پردازنده، به وظیفه^{۱۲} معینی گذارد می‌شود. یک پردازنده ارشد^{۱۳}، سیستم را کنترل می‌کند؛ سایر پردازنده‌ها یا به پردازنده ارشد برای

- 1. failure
- 2. graceful degradation
- 4. tandem
- 5. primary
- 7. checkpoint
- 8. duplication
- 9. symmetric multiprocessing model
- 11. asymmetric multiprocessing
- 13. master

- 3. fault-tolerant
- 6. backup
- 10. identicat
- 12. task

سیستم چند پردازنده، کپی همسانی از سیستم عامل را اجرا می‌کند

نامتنازن یک پردازنده هر طبقه معنی‌علی‌لرز.

دستورالعمل‌ها، نگاه می‌کنند و یا وظیفه از پیش - تعریف شده‌ای را انجام می‌دهند. این طرح، یک ارتباط ارباب^۲ - بردهای را تعریف می‌کند. پردازنده ارباب زمان‌بندی نموده و به پردازنده‌های برد، کار، اختصاص خواهد داد.

مثالی از سیستم چند پردازنده‌ای متقارن، ورژن انکور^۳ یونیکس برای کامپیوتر مالتی مکس^۴ می‌باشد. این کامپیوتر می‌تواند طوری پیکربندی شود که یک دو جین پردازنده را که هر کدام یک کپی از یونیکس را اجرا می‌کنند، به کار گیرد. مزیت این مدل آن است که پراسس‌های بسیاری می‌توانند یکباره اجرا شوند (N پراسس به‌ازاء N، CPU) بدون این که سبب کاهش کارآیی گردد. اما، باستی دقیقاً اعمال O/I کنترل شوند تا داده به پردازنده مناسب برسد. و نیز، چون CPU‌ها مجرأ هستند، ممکن است یکی بیهوده بماند در حالی که دیگری زیادی - بار^۵ گردد که نتیجه، نقص بهره‌وری است. برای اجتناب از این عدم بهره‌وری، پردازنده‌ها می‌توانند ساختمان‌های داده مشخصی را به اشتراک گذارند. سیستم چند پردازنده‌ای از این نوع، اجازه خواهد داد که کارها و منابع به صورت پویا مابین پردازنده‌های متعدد تقسیم شوند و سبب کاهش واریانس^۶ در سیستم‌ها خواهد بود. به هر حال، چنین سیستمی باستی به دقت نوشته شود، همان‌گونه که در فصل ۶ خواهیم دید.

چند پردازنده‌های نامتقارن، در سیستم‌های بسیار بزرگ که در آنها یکی از زمان‌گیرترین فعالیت‌ها، پردازش O/I است، معمول‌تر است. در سیستم‌های قدیمی دسته‌ای، پردازنده‌های کوچک، که در مسافتی دورتر از CPU اصلی قرار داشته‌اند جهت راندن کارت خوان‌ها و چاپگرهای خطی و انتقال کارها از و به کامپیوتر اصلی، به کار می‌رفتند. این مکان‌ها، سایت ورودی - کار - دور^۷ (RJES) نامیده می‌شدند. در یک سیستم اشتراک - زمانی، یک فعالیت اصلی I/O، پردازش ورودی / خروجی کرکترها مابین ترمینال‌ها و کامپیوتر، است. اگر کامپیوتر اصلی، می‌بایست برای هر کرکتر هر ترمینال، وقفه^۸ دریافت کند، در این صورت باید تمام اوقاتش را فقط به پردازش کرکترها صرف کند. جهت اجتناب از این وضعیت، اکثر سیستم‌ها یک پردازنده خط - مقدم^۹ جداگانه دارند که کلیه ترمینال O/I را سرویس می‌دهد. به عنوان مثال، یک سیستم بزرگ IBM، باستی یک مینی کامپیوتر Series/1 را به عنوان خط مقدم به کار برد. خط - مقدم نظیر یک بافر مابین CPU اصلی و ترمینال‌ها عمل می‌کند و اجازه می‌دهد که پردازنده اصلی به خطوط یا بلوک کرکترها، به جای کرکترهای منفرد، دسترسی داشته باشد. چنین سیستم‌هایی از قابلیت^{۱۰} اعتماد کم، رنج می‌برند.

1. relationship

2. master-slave

3. encore's version of unix

4. multimax

6. variance

7. remote-job-entry site

5. over load

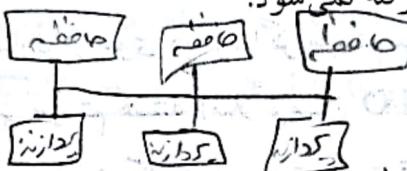
9. front-end

10. reliability

8. interrupt

مهم است تشخیص دهیم که تفاوت مابین چند پردازنده‌های متقارن و نامتقارن، ممکن است حاصل نرم افزار یا سخت افزار باشد. سخت افزار ویژه می‌تواند وجود داشته باشد تا پردازنده‌های چندگانه را تمیز دهیم، یا نرم افزار می‌تواند طوری نوشته شود که تنها یک پردازنده ارباب و چند پردازنده برد مجاز باشد. به عنوان نمونه، سیستم عامل SunOS، ورژن ۴، پردازش نامتقارن و ورژن ۵، Solaris 2، پردازش متقارن را فراهم می‌سازد.

همان‌گونه که مایکرو کامپیوترها ارزان‌تر و قوی‌تر می‌گردند، توابع سیستم عامل اضافی به پردازنده‌های برد، یا پشت‌خط^۱ به صورت خارجی^۲ بار می‌شود. به عنوان مثال، نسبتاً آسان است که مایکرو پراسسوری با حافظه خودی اضافه نماییم تا سیستم دیسک را مدیریت نماییم. مایکرو پراسسور می‌تواند یک توالی از درخواست‌ها از CPU اصلی دریافت نموده و صفحه دیسک خود و الگوریتم زمان‌بندی خود را پیاده‌سازی نماید. این تشکیلات پردازنده اصلی را از بارسر زمان‌بندی دیسک، فارغ می‌سازد. PC‌ها دارای مایکرو پراسسوری در صفحه کلید می‌باشند که ضربه‌های کلید را به کد آنها تبدیل نموده و به CPU می‌فرستند. در واقع، این نحوه کاربرد ریز پردازنده‌ها، به قدری رایج شده است که دیگر به عنوان چند پردازنده‌ای در نظر گرفته نمی‌شود.



1.7. Distributed Systems

۷.۱۰ سیستم‌های توزیع شده

تمایل اخیر سیستم‌های کامپیوتری، توزیع محاسبات مابین پردازنده‌های متعدد است. در مقایسه با سیستم‌های با کوپل‌از سخت که در بخش ۶.۱ بحث شد، پردازنده‌ها حافظه یا کلاک مشترکی ندارند. بلکه هر پردازنده حافظه محلی خود را دارد. پردازنده‌ها از طریق خطوط مخابراتی مختلفی نظیر گذرگاه‌های سریع یا خطوط تلفن با یکدیگر ارتباط برقرار می‌کنند. این سیستم‌ها، معمولاً به نام سیستم‌های با کوپل‌از نرم^۳، یا سیستم‌های توزیع شده رجوع می‌شوند. در سیستم‌های توزیع شده، پردازنده‌ها می‌توانند در اندازه و عمل متفاوت باشند. آنها ممکن است شامل مایکرو پراسسورهای کوچک، استگاه‌های کاری، مینی کامپیوترها، و سیستم‌های کامپیوتری همه‌منظوره بزرگ باشند. این پردازنده‌ها با اسمی گوناگونی مانند سایت‌ها^۴، گره‌ها^۵، کامپیوترها، و غیره رجوع می‌شوند.

دلایل متعددی جهت ساختن سیستم‌های گسترش وجود دارند، که مهمترین آنها عبارتند از:

- ۱. اشتراک منابع. اگر تعدادی از سایت‌های مختلف (با قابلیت‌های مختلف) به یکدیگر متصل شوند، در این صورت کاربری در یک سایت قادر خواهد بود منابع^۶ در دسترس سایت دیگر را به کار گیرد. مثلاً، کاربر



مُحاصَّه سایت‌های توزیع شده با دیگر سایت‌ها

1. off-load
2. back-end
4. sites
5. nodes

3. loosely coupled

سایت A، چاپگر لیزری سایت B را مورد استفاده قرار می‌دهد. بدین وسیله کاربر مرکز B هم به فایل‌های واقع در مرکز A می‌تواند دسترسی داشته باشد. عموماً، اشتراک منابع در یک سیستم گسترده، مکانیزم‌هایی برای اشتراک فایل‌ها در سایت‌های دور، پردازش اطلاعات در بانک‌های اطلاعاتی توزیع شده، چاپ فایل‌ها در سایت‌های دور، استفاده از اجزاء سخت‌افزاری خاص دور (مانند پردازنده آرایه‌ای^۱، و اجرای اعمال دیگر، ارائه می‌دهد.

• تسريع محاسبات. اگر محاسبه ویژه‌ای بتواند به تعدادی زیر محاسبات همروند تجزیه شود، سیستم توزیع شده مجاز می‌دارد که آنها را مابین مراکز مختلف توزیع کنیم - اجرای همروند آن محاسبه. به علاوه، اگر سایت خاصی در حال حاضر از کار زیاد، اورلود شود، تعدادی از کارها می‌توانند به سایت دیگر کم‌بار منتقل شوند. این نقل مکان کارها به نام اشتراک - بار^۲ نامیده می‌شود.

• قابلیت اعتماد. اگر در سیستم توزیع شده، سایتی خراب شود، سایت‌های باقی بالقوه قادر به ادامه عمل هستند. اگر سیستم مرکب از کامپیوترهای همه-منظوره خودمختار^۳ بزرگ (تأسیسات)^۴، باشد، (خرابی یکی از آنها در عملکرد بقیه اثری ندارد. از سوی دیگر، اگر سیستم مرکب از تعدادی ماشین‌های کوچک، که هریک مسئول یک عمل اساسی سیستم هستند (مثلًاً ترمینال I/O کرکتری یا سیستم فایل) باشد، در این صورت یک خرابی ساده، به طور مؤثری کل عمل سیستم را متوقف خواهد ساخت. عموماً اگر به اندازه کافی افرونگی^۵ در سیستم وجود داشته باشد (در هر دو مورد داده و سخت‌افزار)، سیستم قادر به ادامه کار حتی در صورت خرابی تعدادی سایت، خواهد بود.

• ارتباط. نمونه‌های بسیاری وجود دارند که برنامه‌ها لازم است با یکدیگر در یک سیستم، به مبادله داده پردازند. سیستم‌های پنجره‌ای یک مثال زنده است، زیرا آنها غالباً داده را تقسیم می‌کنند یا داده را بین نمایش‌ها انتقال می‌دهند. وقتی سایت‌های بسیاری، از طریق شبکه‌های ارتباطی، به یکدیگر متصل شوند، پراسس‌های سایت‌های مختلف موقعیت تبادل اطلاعات را خواهند داشت. کاربران می‌توانند انتقال فایل‌ها را آغاز کنند یا از طریق پست الکترونیکی^۶ با یکدیگر ارتباط داشته باشند. کاربری می‌تواند به کاربر دیگر در همان سایت یا سایت دیگری، پست بفرستد.

1. array-processor

4. redundancy

2. load-sharing
5. installations

3. autonomous
6. electronic mail

۱.۸. سیستم‌های بی‌درنگ

۱.۸.۱. Real-Time Systems

فرم دیگری از سیستم‌های عامل همه-منظوره، سیستم‌های بی‌درنگ می‌باشند. سیستم بی‌درنگ، زمانی که نیاز زمانی صلب^۱ در عمل پردازنده یا جریان داده وجود داشته باشد، به کار می‌رود. و لذا معمولاً به عنوان یک وسیله کنترلی در یک کاربرد اختصاصی^۲، عمل می‌نماید. سنسورها داده را به کامپیوتر وارد می‌کنند. کامپیوتر باید داده را تحلیل نموده و احتمالاً کنترلی برای تغییر ورودی‌های سنسور، تنظیم نماید. سیستم‌هایی که آزمایشات علمی را کنترل می‌کنند، سیستم‌های تصویری پزشکی، سیستم‌های کنترل صنعتی، و بعضی سیستم‌های نمایشی نمونه‌هایی از سیستم‌های بی‌درنگ می‌باشند. همین طور سیستم‌های تزریق سوت موتور اتومبیل، کنترل کننده‌های لوازم خانگی و سیستم‌های تسليحاتی، مشمولند. سیستم بی‌درنگ، محدودیت‌های^۳ زمانی ثابت و خوش تعریفی را دارد. پردازش، باید بر طبق محدودیت تعریف شده‌ای انجام گیرد، در غیر این صورت سیستم از کارمی افتاد. به عنوان مثال، یک بازوی ربات باید بعد از تصادم درون ماشینی که در حال ساختش است، فرمان توقف دریافت کند. سیستم بی‌درنگ تنها وقتی درست کار می‌کند که نتایج بازگشتی را طی زمان محدودی ارائه دهد. این وضعیت را با سیستم اشتراک-زمانی مقایسه نماید که در آن مطلوبست (ولی اجباری نیست) که زمان پاسخ سریع باشد، یا با یک سیستم دسته‌ای، که در آن هیچ محدودیت زمانی، مفروض نیست.

۱.۸.۲. محدودیت زمانی

سیستم‌های بی‌درنگ، بر دو گونه‌اند. سیستم بی‌درنگ سخت، تضمین می‌کند که وظایف بحرانی^۴، به موقع کامل می‌شوند. این هدف لازم می‌دارد که تمام تأخیرها در سیستم، از زمان بازیابی داده ذخیره شده تا زمانی که سیستم عامل فرمانی را بر روی آن به اتمام می‌رساند، محدود باشند. چنین محدودیت‌های زمانی، سهولت‌هایی^۵ را که در سیستم بی‌درنگ سخت وجود دارد، محرز می‌سازد. ذخیره ثانویه از هر نوع، معمولاً، محدود یا حذف است، و داده به جای آن در حافظه کوتاه مدت^۶ یا در حافظه فقط خواندنی (ROM) قرار می‌گیرد. ROM به عنوان حافظه غیر میرا، در صورت قطع برق داده را نگه می‌دارد. اکثر ویژگی‌های سیستم‌های عامل پیشرفت، نیز غایبند، زیرا آنها تمایل به جداسازی کاربر از سخت‌افزار دارند و این جدایی منجر به عدم قطعیت زمان اجرای یک عمل، می‌گردد. به عنوان نمونه، حافظه مجازی، تقریباً هرگز در سیستم‌های بی‌درنگ یافته نمی‌شود. لذا، سیستم‌های بی‌درنگ سخت با عمل سیستم‌های اشتراک-زمانی، تناقض دارند و این دو نمی‌توانند توأمًا وجود داشته باشند، از آنجایی که هیچ یک از سیستم‌های عامل همه-منظوره موجود، تابعیت سیستم عامل بی‌درنگ سخت، را حمایت نمی‌کنند، در این متن به این

1. rigid

4. critical tasks

2. dedicated

5. facilities

3. constraint

6. short-term memories

نوع سیستم، توجه نمی‌کنیم.

یک نوع محدود از سیستم‌های بی‌درنگ، سیستم بی‌درنگ نرم نامیده می‌شود که در آن یک وظیفه بی‌درنگ بحرانی، نسبت به سایر وظایف، اولویت دارد و تا پایان تکمیل شدن این ارجحیت را نگه می‌دارد. همانند سیستم‌های بی‌درنگ سخت، تأخیر زمانی‌های هسته^۱ لازم است محدود شود. یک وظیفه بی‌درنگ نرم یک هدف قابل رسیدن است و میسر است که با سایر انواع سیستم‌ها مخلوط گردد. اما، سیستم‌های نرم، یوتیلیتی‌های کمتری نسبت به سیستم‌های سخت دارند. با درنظر داشتن عدم حمایت مهلت زمانی^۲، استفاده از آنها در کنترل صنعتی و رباتیک، ریسک‌آور است. فیلدهای متعددی وجود دارند که می‌توان از آنها استفاده نمود: شامل چند رسانه‌ای^۳، واقعیت مجازی^۴، و پروژه‌های علمی پیشرفته مانند تجسس زیردریاها و سیارات. این سیستم‌ها به ویژگی‌های سیستم‌های عامل پیشرفته که توسط سیستم‌های بی‌درنگ سخت حمایت نمی‌شدند، نیازمندند. به دلیل استفاده گسترده از قابلیت^۵ عمل سیستم بی‌درنگ نرم، این سیستم راه خود را در سیستم‌های عامل جاری، شامل نسخه‌های مهم یونیکس باز کرده است. در فصل^۶، ما سهولت زمانبندی لازم برای پیاده‌سازی قابلیت عمل بی‌درنگ نرم در یک سیستم عامل را بررسی می‌کنیم. در فصل^۷، طراحی مدیریت حافظه را برای محاسبات زمان-حقیقی^۸ تشریح می‌کیم.

۹.۱ خلاصه

1.9. Summary

سیستم‌های عامل طی ۴۰ سال اخیر به منظور دو هدف اصلی، توسعه یافته‌اند. اول، سیستم‌های عامل سعی دارند فعالیت‌های محاسباتی را طوری زمانبندی نمایند که کارآیی بالایی برای سیستم کامپیوتر، حاصل شود. دوم، آنها محیط راحتی برای توسعه و اجرای برنامه‌ها فراهم می‌آورند.

در ابتدا، سیستم‌های کامپیوتری از کنسول-مقدم^۹، مورد استفاده قرار می‌گرفتند. نرم‌افزارها از قبیل اسپلرها، بارکتنددها، اتصال دهندها، و کامپایلرها سهولت برنامه‌سازی سیستم را بهبود بخشیدند، اما البته، نیازمند زمان برپایی^{۱۰} دسته^{۱۱} کردند.

سیستم‌های دسته‌ای، توالی خودکار کارها را توسط سیستم عامل مقیم، مجاز نموده و بهره‌وری کلی کامپیوتر را

- | | | |
|--------------------|------------------|---------------|
| 1. kernel | 2. deadline | 3. multimedia |
| 4. virtual reality | 5. functionality | 6. real-time |
| 7. front console | 8. set-up | 9. facilities |
| 10. batched | | |

بپردازند. کامپیوتر، دیگر منتظر عمل انسانی نمی‌ماند. اما، بهره‌وری CPU هنوز پایین بود، زیرا، سرعت عمل و سایل I/O قابل مقایسه با CPU نبود. عمل خارج-خط دستگاه‌های کند، وسیله‌ای برای استفاده از چندین سیستم کارت خوان-به-نوار و نوار-به-چاپگر برای یک CPU، فراهم آورد. اسپولینگ، اجازه می‌داد که CPU ورودی بی کار را با محاسبه و خروجی کارهای دیگر، روی هم بیاندازد.

جهت افزایش کارآیی کلی سیستم، توسعه دهنده‌گان، مفهوم چند برنامه‌ای را معرفی کردند. در چند برنامگی، کارهای متعددی در یک زمان در حافظه نگه داشته می‌شوند؛ CPU بین آنها، پس و پیش، سوئیچ می‌کند تا میزان استفاده CPU را افزایش داده و زمان کلی لازم برای اجرای کارها را، کاهش دهد.

چند برنامگی، که برای بهبود بازدهی توسعه یافته بود، همین‌طور، اشتراک-زمانی را مجاز نمود. سیستم‌های اشتراک-زمانی اجازه می‌دهند کاربران بسیاری (از یک تا چند صد) در آن واحد، از سیستم کامپیوتی به طور مجاوره‌ای استفاده نمایند.

سیستم‌های کامپیوتی شخصی، ریزکامپیوتراهایی هستند که به طور قابل توجهی کوچک‌تر و کم‌هزینه‌تر از سیستم‌های مین فریم می‌باشند. سیستم‌های عامل این ماشین‌ها به طرق وسیعی از توسعه سیستم‌های عامل مین فریم، بهره جسته‌اند. اما، از آنجاکه افراد، استفاده منفرد از سیستم دارند، بهره‌وری CPU دیگر حرف اول نیست. لذا بعضی از تصمیمات طراحی در سیستم‌های عامل مین فریم، در این سیستم‌های کوچک‌تر، مناسب نیستند.

سیستم‌های موازی بیش از یک پردازنده در ارتباط نزدیک با هم، دارند؛ پردازنده‌های مرکزی گذرگاه کامپیوترا به اشتراک گرفته و گاهی حافظه و وسایل جانبی را نیز تقسیم می‌کنند. چنین سیستم‌هایی، افزایش توان عملیاتی و نیز قابلیت اعتماد کاملی را فراهم می‌سازند.

سیستم‌های توزیع شده مجموعه‌ای از پردازنده‌ها هستند که حافظه یا ساعتی را به اشتراک نگرفته‌اند. به جای آن، هریک از پردازنده‌ها حافظه محلی خود را دارند و با سایر پردازنده‌ها از طریق خطوط مخابراتی متعددی نظیر گذرگاه‌های سریع یا خطوط تلفن، ارتباط برقرار می‌کنند. سیستم توزیع شده، برای کاربر، دسترسی به منابع متعدد واقع در سایت‌های دور را، فراهم می‌آورد.

یک سیستم بی‌درنگ سخت، غالباً به عنوان یک وسیله کنترلی در یک کاربرد اختصاصی استفاده می‌شود. سیستم بی‌درنگ سخت، محدودیت‌های زمانی ثابت و خوش تعریفی را دارد. پردازش باید در چهارچوب محدودیت تعريف شده، انجام گیرد، در غیر این صورت، سیستم از کار می‌افتد. سیستم‌های بی‌درنگ نرم، کمتر محدودیت زمانی جدی دارند و زمان‌بندی مهلت دار را ساپورت نمی‌نمایند.

ما، پیش روی منطقی توسعه سیستم عامل را نشان داده‌ایم و این که شامل بودن ویژگی‌ها در سخت‌افزار CPU که

لازم است کار سیستم عامل پیش فته است، در این سیر مؤثر بوده است. این تمايل، امروزه می تواند در تکامل کامپیوترهای شخصی، با سخت افزار کم هزینه رو به رشد برای مجاز ساختن مشخصات توسعه یافته، به نوبه خود، مشاهده شود.

■ تمرینات

۱. سه هدف اصلی یک سیستم عامل چیست؟

۲. چهار قدم ضروری در اجرای یک برنامه بر روی یک سیستم کاملاً اختصاصی را نام ببرید.

۳. یک روش حدی اسپولینگ، به نام، مرحله‌ای^۱ کردن نوار، آن است که کل نوار مغناطیسی را بر روی دیسک، قبل از استفاده کپی کنیم. بهره اصلی چنین طرحی را مورد بحث قرار دهید.

۴. در یک محیط چند برنامگی و اشتراک - زمانی، کاربران متعددی همزمان سیستم را تقسیم می کنند. این وضعیت منجر به مسائل امنیتی متعددی می گردد.

a. دو تا از این مسائل را نام ببرید.

b. آیا می توان در یک ماشین اشتراک - زمانی، به اندازه یک ماشین اختصاصی، درجه امنیت را تضمین نمود؟ پاسخ خود را شرح دهید.

۵. منفعت اصلی چند برنامگی چیست؟^۲

۶. تفاوت های عمدۀ سیستم های عامل کامپیوترهای مین فریم و شخصی چیستند؟^۳

۷. خاصیت اساسی سیستم های عامل از انواع زیر را تعریف کنید:

a. دسته ای c. اشتراک - زمانی e. توزیع شده b. محاوره ای d. بی درنگ

۸. بیان کرده ایم که سیستم عامل برای استفاده سودمند از سخت افزار کامپیوتر، ضروری است. چه زمانی مناسب است که سیستم عامل این اصل را رها کند و منابع را هدر دهد؟ چرا چنین سیستمی واقعاً زیان آور نیست؟

۹. تحت چه شرایطی بهتر است کاربری یک سیستم اشتراک - زمانی را، off-use کند، به جای آن که یک کامپیوتر شخصی یا ایستگاه کاری تک - کاربره را به کار گیرد.

۱۰. تفاوت های چند - پردازشی متقارن و نامتقارن را تشریح کنید. سه نفع و یک ضرر سیستم چند پردازندگی را نام ببرید.

۱۱. چرا سیستم های توزیع شده، مطلوبند؟

۱۲. مشکل اصلی یک برنامه نویس در نوشتن سیستم عامل برای محیط بی درنگ، چیست؟

1. staging