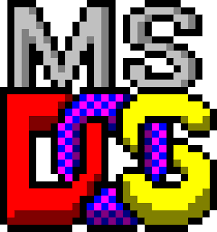
**سیستم عامل**

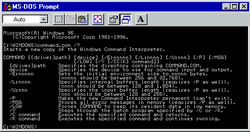
**انواع سیستم عامل از نظر ساختاری**

**یکپارچه (ساده)**

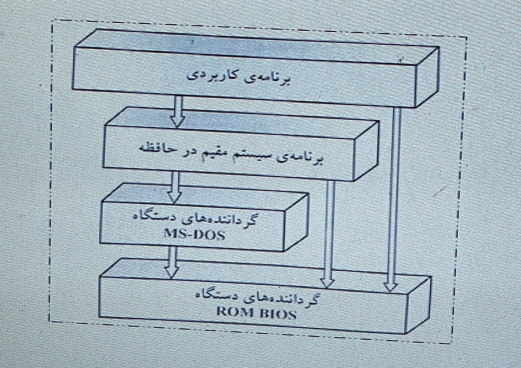
ساده ترین ساختار برای سیستم عامل است در این ساختار واسط ها و سطوح عملکرد به خوبی از هم تفکیک نشده اند و برنامه های کاربردی می توانند به روال i/o(ورودی و خروجی) دستیابی داشته باشند و مستقیما بر روی مانیتور یا دیسک بنویسند

اغلب سیستم عامل های تجازی. ساختار های خوش تعریفی ندارند غالبا چنین سیستم هایی به صورت سیستم هال کوچیک ساده و محدود شروع به کار می کنند و سپس نسبت به حوزه اصلی خود رشد می کنند





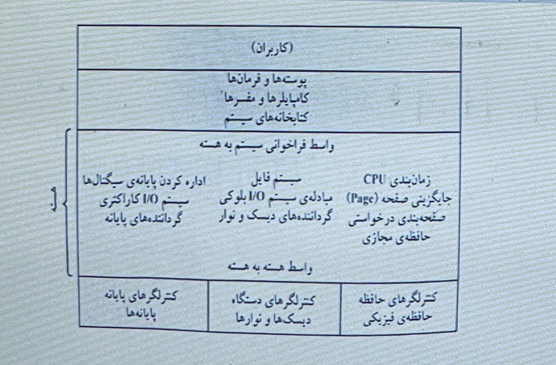
MS DOS مثالی از چنین سیستمی است . این سیستم عامل نوشته شده تا بیشترین قابلیت را در کمترین فضا فراهم سازد و در نتیجه به دقت به پیمانه ها تقسیم نشده است



در MS-DOS واسط ها و سطوح عملکرد به خوبی تفکیک نشده اند برای مثال برنامه های کاربردی قادرند به روال های i/o پایه نیز دستیابی داشته باشند یا مستقیما در نمایشگر و گرداننده های دیسک بنویسند این آزادی ms-dos را در مقابل برنامه های مضر آسیب پذیز می سازد و در نتیجه موجب از کار افتادم سیستم می شود

چون در intel 8088 که سیستم عامل MS-Dos برای آن نوشته شد فاقد حالت دوگانه و فاقد حفاظت سخت افزاری است طراحان ms-dos هیچ انتخابی نداشتند و سخت افزار را دسترس پذیر رها کردند

مثال دیگر ساختار محدود سیستم عامل یونیکس اولیه است همانند MS-DOS یونیکس ابتدا توسط امکانات سخت افزاری محدود شد یونیکس شامل دو بهش بود هسته (کرنل - kernel) و برنامه های سیستم . هسته به مجموعه ای از واسط ها و گرداننده های دستگاه تبدیل شد که طی چندین سال اضافه شدند و بسط یافتند . سیستم عامل های قدیمی یونیکس را می توان لایه ای در نظر گرفتدر شکل زیر امده است هر چیز موجود در در پایین واسط فراخوان سیستم و بالای سخت افزار فیزیکی هسته است

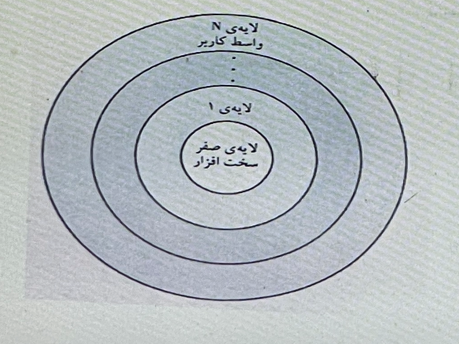


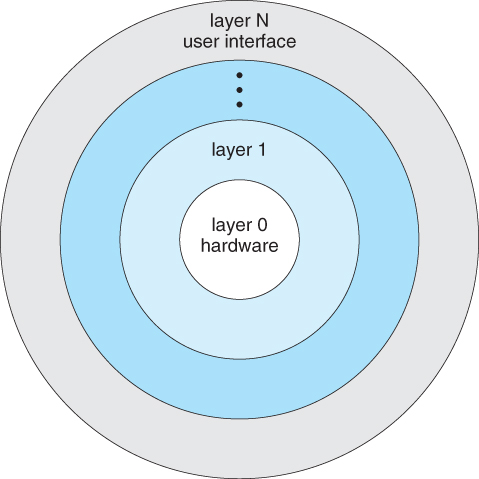
هسته از طریق فراخوان های سیستمی سیستم فایل زمانبندی پردازنده مدیریت حافظه و سایر سیستم عامل را فراهم می سازد .جمع شدن امکانات (عملکرد های) زیاد در یک سطح اشتباه است پیاده سازی و نگهداری این ساختار یکپارچه دشوار بود هر چند که امتیاز خاصی نیز داشت: در واسط فراخوان سیستم یا در ارتباطات داخل هسته سرباز بسیار کمی وجود دارد

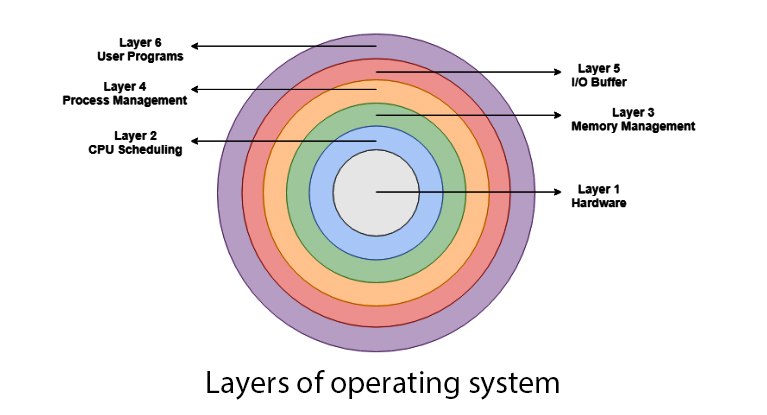
**لایه ای**

با پشتیبانی سخت افزاری مناسب سیستم عامل ها می توانند به مولفه هایی تبدیل شوند که نسبت به سیستم عامل های یونیکس و ms-dos اولیه کوچک تر و مناسب تر باشند در این صورت سیستم عامل روی کامپیوتر و برنامه های کاربردی که از آن استفاده می کنند کنترل خیلی بیشتری دارد پیاده ساز ها, آزادی بیشتری در تغییر عملکرد داخلی سیستم و ایجاد سیستم عامل های پیمانه ای دارند **پنهان سازی اطلاعات به پایین** . ویژگی ها و عملکرد کلی تعیین به مولفه هایی تقسیم می شوند **پنهان سازی اطلاعات** نیز مهم است

سیستم عامل به چند لایه (سطح ) تبدیل می شود لایه پایینی (لایه شماره0 ) سخت افزار است لایه ی بالایی (لایه شماره N) واسط کاربر استاین ساختار لایه ای در شکل زیر امده است







نمونه ای از لایه ی سیستم عامل – مثلا لایه ی M شامل ساختمان داد و مجموعهای از روال ها است که میتواند توسط لایه های سطج بالاتر فراخوانی شود . لایه ی M به نوبه خود می تواند عملیات روی لایه های سطح پایین تر را فراخوانی نماید

**امتیاز اصلی روش لایه ای سهولت ساخت و اشکال زدایی است**

لایه ها طوری انتخاب میشوند که خر کدام از توابع (عملکرد ها) و سرویس های لایه های پایین تر استفاده می کنند این روش اشکال زدایی و وارسی (Cerification) سیستم را آسان می سازد.

اولین لایه می تواند بدون توجه به بقیه ی سیستم اشکال زدایی شودزیر طبق تعریف فقط از سخت افزار (که فرض می شود درست است) برای پیاده سازی توابه خود استفاه می کند وقتی اولین لایه اشکال زدایی شد می توان فرض کرد که به درستی عمل می کند و لایه ی دوم اشکال زدایی شود اگر خطایی در اثنای اشکال زدایی لایه خاصی زخ دهد خطا باید در آن لایه باید . زیرا لایه های زیر آن قبلا اشکال زدایی شدند

بنابراین این طراحی و پیاده سازی سیستم، ساده است.

هر لایه فقط توسط عملکردهایی پیاده سازی می شود که توسط لایه های پایین تر فراهم شده اند. لازم نیست یک لایه بداند که این عملکرد ها چگونه پیاده سازی شده اند.

مشکل عمده ی روش لایه ای تعریف مناسب لایه های گوناگون است

مشکل دیگر پیاده سازی لایه های این است که نسبت به انواع دیگر کارایی کمتری دارند

برای مثال ، وقتی برنامه ی کاربر یک عمل i/o را انجام می دهد، فراخوان سیستمی را اجرا می کند که در خود لایه i/o دچار تله می شود که لایه مدیریت حافظه را فراخوانی می کند که آن نیز به نوبه ی خود ، لایه ی زمانبند پردازنده را فراخوانی می کند که سپس به سخت افزار ارسال می شود. درهر لایه ، پارامتر ها ممکن است تغییر کند ممکن است نیاز به ارسال داده باشد . هر لایه سربازی را به فراخوان سیستم اضافه می کند نتیجه اش این است که فراخوان سیستمی در روش لایه ای نسبت به سیستم غیر لایه ای بیشتر طول می کشد

**ریز هسته ای یا مایکرو کرنل ها**

این روش ، سیستم عامل را با حذف تمام مولفه های غیر اساسی از هسته و پیاده سازی آن به صورت برنامه های سطح کاربر و سیستمی ، سازماندهی می کند. نتیجه، هسته ی کوچک تری است. معمولا ریز هسته ها، علاوع بر تسهیلات ارتباطی، کمتریسن مدیریت حافظه و فرایند را فراهم می کنند

در اواسط دهه 1980، پژوهشگران در دانشگاه Carnegie Mellon، سیستم عاملی را به نام Mach را تولید کردند، که هسته را با استفاده از، روش ریز هسته پیمانه ای کرده است

**پیمانه ها (ماژول ها)**

شاید بهترین فناوری برای طراحی سیستم عامل شامل استفاده از پیمانه های بازشدنی هسته باشد. این نوع طراحی در پیاده سازی های مدرن یونیکس مثل سولاریس، لینوکس و Mac OS X و ویندوز متداول است.

ایده ی این طراحی این است که هسته سرویس های اصلی را ارایه کنددر حال که سرویس های دیگر به طور پویا پیاده سازی می شوند. لینک کردن به سرویس ها به طور پویا، ویژگی های جدیدی را مستقیما به هسته اضافه می کند که لازم است هر وقت هسته تغییر می کند دوباره کامپایل شود.

نتیجه ی کلی شبیه یک سیستم لایه ای است که هر بخش هسته دارای واسط های تعریف شده و حفاظت شده است; وقتی از سیستم لایه ای انعطاف پذیرتر است. زیرا هر پیمانه می تواند هر پیمانه دیگری را فراخوانی کند. این روش شبیه ریز هسته نیز هست. زیرا پیمانه اصلی تنها وظایف اصلی را بر عهده دارد و می داند که پیمانه های دیگر را چگونه بار و با آن ها ارتباط بر قرار نماید. اما کارامدتر است. زیر پیمانه ها

برای برقراری ارتباط نیاز به مبادله ی پیام ندارند.

**هفت نوع پیمانه بارشدنی هسته در سیستم عامل سولاریس:**

1. **کلاس هیای زمان بندی**
2. **سیستم های فایل**
3. **فراخوان های سیستمی باز شدنی**
4. **فرمت های قابل اجرا**
5. **پیمانه های streams**
6. **پیمانه های متفرقه (گوناگون)**
7. **گرداننده های دستگاه و گذرگاه (باس)**

**ماشین مجازی (Virtual Machine)**

یک سیستم کامپیوتری دارای لایه های سخت افزار