فصل اول تئوری مفاهیم سیستم عامل

یک سیستم کامپیوتری از بخش‌های زیر تشکیل می‌شود:

1- سخت افزار 2- نرم افزار 3-برنامه‌های کاربردی

نرم افزارها نقش مهمی در یک سیستم کامپیوتری دارند. سیستم عامل یک نرم افزار است. نرم افزارها به دو دسته تقسیم می‌شوند:

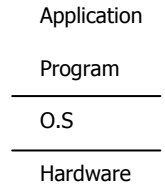
1- نرم افزارها یا برنامه سیستمی – برای کارهای سیستمی عملیات کامپیوتر را بر روی خودشان کنترل می‌کنند. بصورت مستقیم با کاربران کار نمی‌کنند.
2- برنامه‌های کاربردی – جنبه‌های کاربردی دارند و مشکلات مستقیمی از کاربران را حل می‌کنند.

سیستم عامل نرم افزاری است که جز دسته سیستمی‌ها قرار می‌گیرد یک نرم افزار سیستمی است. سیستم عامل هیچ المان سخت افزاری ندارد. حتی اگر سخت افزارهایی باشند که فقط سیستم عامل از آن استفاده می‌کند آن سخت افزار بخشی از سیستم عامل نیست.

تعاریفی از سیستم عامل

- کنترل کننده تمام منابع کامپیوتر و پایه‌ای برای تهیه برنامه‌های کاربردی می‌باشد.
- سیستم عامل می‌تواند پایه‌ای باشد که برنامه‌های کاربردی بر روی آن کار کنند. منابع یک سیستم مانند زمان CPU ، فضای حافظه ، فضای ذخیره فایل و دستگاههای I/O
- سیستم عامل روی سخت افزار قرار می‌گیرد و باعث می‌شود پیچیدگی دستگاه‌های سخت افزاری از دید کاربر مخفی باشد.

- وظیفه اصلی سیستم عامل پنهان کردن پیچیدگی سیستم و ارائه یک مجموعه مناسب به برنامه نویس برای کار بر روی آن می‌باشد.



برنامه‌های سیستمی دیگر
- Compiler
- Linker
- Loader
- Editor
- Command Interpreter

این برنامه‌ها بخشی از سیستم عامل نیستند زیرا اگر جزئی از سیستم عامل می‌بودند نباید قابل تغییر و قابل انتخاب باشند. مثلا نمی‌توان مدیریت حافظه سیستم عامل را عوض کرد. ولی کامپایلر قابل تغییر است. عناصر داخل سیستم عامل ماهیت یک سیستم عامل را تشکیل می‌دهند. کرنل در مورد گرافیکی بودن یا نبودن معنی ندارد این موضوع در مورد Command Interpreter است. این موضوع که سیستم عامل ویندوز یک سیستم عامل گرافیکی است غلط است.

- دو دید مختلف در مورد سیستم عامل
  - سیستم عامل همانند یک ماشین توسعه یافته است
    Extended Machine
  - سیستم عامل همانند مدیر منابع
    Resource Manager

سه هدف اصلی سیستم عامل

1- سهولت استفاده از کامپیوتر
2- کارآمدی – یعنی از امکانات بصورت بهینه استفاده کند.
3- قابلیت رشد – یعنی بتوان به راحتی سیستم عامل را رشد و ارتقاء داد.

سیستم‌های محاوره‌ای و دسته‌ای

سیستم‌ها از جهت نحوه‌ی ارتباط با کاربر به دو دسته‌ی «سیستم‌های محاوره‌ای» و «سیستم‌های دسته‌ای» تقسیم می‌شوند.

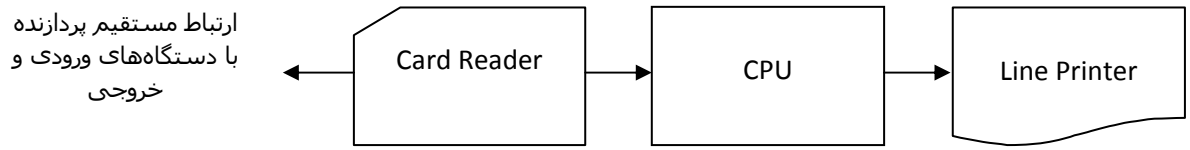
سیستم‌های محاوره‌ای – سیستم‌هایی هستند که در آنها کاربر بطور مستقیم و روی خط (On – Line) با کامپیوتر در ارتباط است. کاربر دستوراتی را وارد می‌کند و منتظر پاسخ می‌ماند و پس از دریافت پاسخ، مجددا دستوراتی را وارد می‌کند.

سیستم‌های دسته‌ای – سیستم‌هایی هستند که در آنها دریافت دستورات یا برنامه‌های کاربر و سپس اجرای آنها در دو مرحله صورت می‌گیرد. ابتدا برنامه‌هایی که عموما دارای نیازهای مشابه نظیر کامپایلر هستند یک گروه به سیستم وارد شده و پس از باز شدن کامپایلر مورد نیازشان، اجرای آنها بطور متوالی آغاز می‌شود.

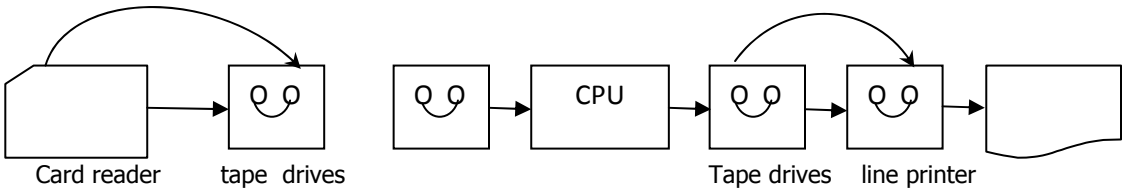
سیستم‌های on – line و off – line

سیستم‌ها از جهت ارتباط با دستگاه‌های جانبی، در دو دسته‌ی سیستم‌های on – line و off – line قرار می‌گیرند.

سیستم‌های on – line با ارتباط مستقیم – در این سیستم‌ها پردازنده مستقیما با دستگاه‌های ورودی و خروجی در ارتباط است و بدلیل کند بودن این دستگاه‌ها کارائی پردازنده به میزان قابل توجهی کاهش می‌یابد.



سیستم‌های off – line با ارتباط غیر مستقیم – در این سیستم‌ها، پردازنده بطور مستقیم با دستگاه‌های ورودی و خروجی کند در ارتباط نیست. ابتدا عمل خواندن ورودی توسط کارت خوان‌های مستقل از سیستم اصلی انجام گرفته و ورودی به حافظه‌های جانبی (نوار مغناطیسی) که سریع‌تر هستند منتقل می‌شود، سپس این نوارها در سیستم اصلی استفاده شده، ارسال خروجی نیز بر روی نوار مغناطیسی انجام می‌گیرد و نهایتا در محل دیگری بر روی چاپگرها ارسال می‌شود، در چنین حالتی امکان همپوشانی I/O و کار پردازنده وجود دارد.



ارتباط غیرمستقیم پردازنده با دستگاه‌های ورودی و خروجی

بافر کردن (Buffering)

علی‌رغم استفاده از نوارهای مغناطیسی باز هم عملیات ورودی و خروجی کند بود، بهره‌وری سیستم را کاهش می‌داد. با استفاده از حافظه‌های میانی (بافرها) عملیات ورودی و خروجی یک برنامه با اجرای آن همزمان انجام می‌شود البته این

روش برای کارهائی که حجم محاسبات ورودی و خروجی متناسبی داشته باشد مفید است.

کارها
- کارهای I/0 bound (I/O Limited)، کارهایی که بخش زیادی از اجرای آنها در ارتباط با دستگاه‌های ورودی / خروجی بوده و محاسبات زیادی ندارند.
- کارهای CPU bound (CPU Limited) کارهایی که حجم زیادی محاسبات داشته و بخش عمده نیاز آنها برای اجرا، دقت پردازنده است.

در این دو نوع کار استفاده از بافر تاثیر زیادی در بهبود اجرا ندارد.

Spooling

در این زمینه از تکنولوژی دیسک استفاده شده است. در این روش کارت‌ها مستقیما وارد دیسک می‌شوند و دیگر نیازی به برقراری ارتباط مستقیم با دستگاه کارت خوان نیست. مکان ذخیره‌ی کارت‌ها در جدول توسط سیستم عامل ثبت می‌شود. وقتی کاری اجرا می‌شود سیستم عامل درخواست‌های آن کار را برای ورودی از دستگاه کارتخوان و از روی دیسک تامین می‌کند به همین ترتیب زمانی که کار برای خروج یک خط چاپگر را صدا می‌زند و خروجی در بافر کپی شده و در دیسک نوشته می‌شود وقتی که کار تکمیل گردید، خروجی چاپ می‌شود. در این روش دیسک مانند یک بافر وسیع برای خواندن هر چه بیشتر از دستگاه ورودی و نگهداری فایل‌های خروجی فرض می‌شود.

چند برنامه‌گی

با استفاده از عمل بافر کردن و Spooling می‌توان چند کار را بطور همزمان اجرا کرد. «چند برنامه‌گی» اجرای یک کار تا زمان نیاز به ورودی یا خروجی ادامه می‌یابد،

سپس وقتی زمان انجام ورودی یا خروجی آن شده پردازنده اجرای کار دیگری را شروع کرده یا ادامه می‌دهد. با این امکان می‌توان با زمان‌بندی کارهای Job Scheduling و استفاده از الگوریتم‌های مختلف زمان اجرای چند کار را بهینه نمود.

اشتراک زمان یا چند وظیفه‌ای

این روش تعمیمی منطقی از روش چند برنامه‌گی است. در این روش کارهای متعددی توسط سوئیچ کردن CPU بین آنها، اجرا می‌شود. اما سوئیچ کردن‌ها بطور تکراری رخ می‌دهند و کاربر می‌تواند در هنگام اجرا با برنامه محاوره داشته باشند. اشتراک زمان، شکل خاصی از چند برنامه‌گی است که در آن تعویض یک کار بر اساس یک معیار زمانی و نه بر اساس زمان نیاز آن کار به ورودی یا خروجی صورت می‌گیرد. اجرای یک کار تا پایان بازه زمانی ادامه پیدا می‌کند سپس پردازنده به یک برنامه دیگر اختصاص می‌یابد. تعیین بازه‌ی زمانی از پارامترهای سیستم عامل است که در کارائی سیستم موثر است و چون این بازه‌های زمانی کوتاه هستند هر کاربر تصور می‌کند سیستم بطور کامل در اختیار اوست.

سیستم‌های موازی

سیستم‌های چند پردازنده‌ای هستند یک مزیت این سیستم‌ها توان عملیاتی بالایشان می‌باشد. با افزایش پردازنده‌ها کار انجام یافته در یک بازه‌ی زمانی افزایش پیدا می‌کند دلیل دیگر برای استفاده از این سیستم‌ها reliability یا قابلیت اطمینان است. اگر کارها بدرستی بین پردازنده‌ها توزیع شود در صورت خرابی یک پردازنده سیستم متوقف نمی‌شود.

متداول‌ترین سیستم‌های چند پردازنده‌ای ، مدل چند پردازنده متقارن Symmetrical Multi Processing Model است که در آن هر پردازنده کپی یکسانی از سیستم عامل را اجرا می‌کند.

بعضی از سیستم‌ها چند پردازنده تا متقارن را بکار می‌برند asymmetrical Multi Processing در این نوع سیستم‌ها هر پردازنده به وظیفه معینی گمارده می‌شود. یک پردازنده اصلی سیستم را کنترل می‌کند سایر پردازنده‌ها هم یا به پردازنده‌ی اصلی نگاه می‌کنند یا یک وظیفه‌ی از بیش تعیین شده را انجام می‌دهند.

سیستم‌های توزیع شده

سیستم‌های با ارتباط محکم – در این سیستم‌ها پردازنده‌ها دارای پالس ساعت یکسان و حافظه مشترک هستند اجرای کار در آنها مشکل بوده ولی سرعت اجرا بالاست.

سیستم‌های با ارتباط سست – در این سیستم‌ها هر پردازنده پالس ساعت و حافظه‌ی مستقل خود را دارد و سرعت اجرا پایین‌تر از مدل ارتباط محکم است.

دلایل استفاده از این سیستم‌ها

1- اشتراک منابع
2- تسریع محاسبات – اگر یک محاسبه بتواند به چند زیر محاسبه تقسیم شود در این سیستم‌ها این امکان وجود دارد که بر روی پردازنده‌ی مختلف بتوان بطور همزمان محاسبات انجام گیرد.
3- قابلیت اعتماد
4- ارتباط در برخی نمونه‌ها لازم است که برنامه‌ها با هم تبادل داده داشته باشند.

سیستم‌ها بلادرنگ

سیستم‌هایی که در کاربردهای خاصی باید حداکثر در مدت زمان مشخص سرویس دهی لازم را انجام دهند. (زمان پاسخ را گارانتی نمایند) بدلیل نیاز به پاسخ زمانی مناسب در این سیستم‌ها از حافظه‌های مجازی و اشتراک زمانی استفاده نمی‌شود. وقفه‌ها و نحوه‌ی عملکرد سیستم عامل با دستگاه‌های ورودی و خروجی، بدلیل اختلاف سرعتی که بین دستگاه‌های ورودی و خروجی با پردازنده وجود دارد سخت افزار به همراه سیستم عامل باید روشی برای هماهنگ کردن کار این دو قسمت بوجود آورد.

انتظار فعال (Busy Waiting)

1- پردازنده پس از تنظیم دستگاه جانبی از طریق راه اندازش، تقاضای خودش را به دستگاه اطلاع می‌دهد.
2- آماده بودن دستگاه برای شروع اجرای دستور و اجرای دستور بررسی می‌شود (ورودی = خواندن، خروجی = نوشتن)
3- اگر دستگاه جانبی آماده نیست به مرحله‌ی قبل برگردد.
4- در صورتی که خواندن یا نوشتن انجام شد پردازنده می‌تواند عمل ورودی و یا خروجی را ادامه دهد یا با اتمام کار، پردازش را دنبال کند.

ضعف – پردازنده مدت زیادی (کاهش بازدهی پایین) را صرف عمل ورودی و یا خروجی نموده = کاهش بازدهی

وقفه Interrupt

هدف – استفاده بهینه از وقت پردازنده

1- پردازنده از طریق راه اندازی دستگاه جانبی اطلاعات مورد نیاز برای خواندن یا نوشتن را در ثبات‌های دستگاه جانبی قرار داده و تقاضای خود را به دستگاه اطلاع می‌دهد.
2- پردازنده کار خود را ادامه داده و دستگاه نیز کار خود را شروع می‌کند (بدون بروز وقفه)
3- دستگاه جانبی پس از اتمام کارش با ارسال یک سیگنال وقفه به پردازنده اتمام کار را اطلاع می‌دهد.
4- پردازنده اطلاعات را در اولین فرصت از دستگاه گرفته و به کار خود ادامه می‌دهد. وقفه در محیط چند برنامه‌گی بسیار موثر است.

وقفه‌ها
1- خارجی – این وقفه‌ها عموما از دستگاه‌های خارجی ناشی می‌شوند مثل دستگاه‌های ورودی یا خروجی – تایمرهای سیستم یا صفحه کلید.
2- داخلی – ناشی از اجرای دستورات خود برنامه مانند حالت‌های تقسیم بر صفر و یا معین نبودن کد دستور

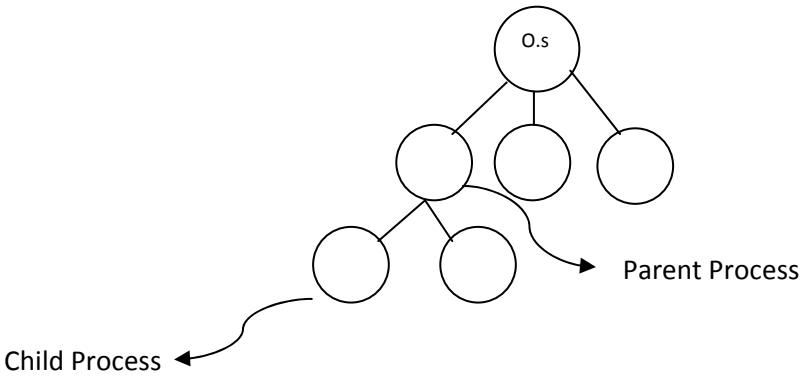
مفهموم پروسس یا پردازه Process

برنامه در حال اجرا همراه با امکاناتی که برای اجرا احتیاج دارد یک پروسس نامیده می‌شود. برنامه یک موجودیت غیرفعال یا Passive است که شامل یکسری فایل ذخیره شده روی دیسک است. اما پروسس فعال یا Active است. برنامه یک موجود زنده نیست ولی پروسس یک موجود زنده است. یک مفهوم کلیدی در تمامی سیستم عامل‌ها پروسس است.

یک پروسس شامل قسمت اجرایی برنامه، اطلاعات برنامه، Dataهای برنامه Stack یا پشته، شماره برنامه، اشاره‌گر به Stack و رجیسترها یا ثبات‌ها و سایر اطلاعات مورد نیاز برای اجرای برنامه می‌باشد.

PCB – اطلاعات مربوط به هر پروسس در جدولی بنام جدول پروسس Process Table یا Process Control Block (PCB) ذخیره می‌شود. یکی از اطلاعات موجود در PCB، Program Counter یا PC است. هر پروسس بغیر از سیستم عامل توسط پروسس دیگری بوجود می‌آید هر پروسس می‌تواند یک یا چند پروسس بوجود بیاورد. سیستم عامل هنگام بوت شدن سیستم بوجود می‌آید.

هر پروسس فقط و حتما یک پروسس پدر دارد و هر پروسس می‌تواند چندین فرزند داشته باشد.



هنگامی که برنامه توسط ما اجرا می‌شود ما از سیستم عامل درخواست می‌کنیم که یک پروسس جدید را بوجود بیاورد.

PID – هر پروسس یک شماره منحصر بفرد دارد بنام PID که سیستم عامل آن پروسس را با آن شماره شناسایی می‌کند. یکی دیگر از چیزهایی که در داخل PCB ذخیره می‌شود همین PID است.

PPID – هر پروسس شماره دیگری نیز دارد که PPID نامیده می‌شود. شماره پروسس پدر خواهد بود که در PCB ذخیره خواهد شد. یعنی پدر پروسس را مشخص می‌کند البته این شماره منحصر بفرد نخواهد بود.

ایجاد فرآیند – مراحل مختلف ایجاد فرآیند عبارتند از:

الف) انتخاب یک کار از بین کارهای موجود سیستم برای تبدیل شدن به فرآیند و اجرای آن که این عمل «زمان‌بندی بلند مدت» (Long term Scheduling) یا «زمانبندی کار» (Job Scheduling) نام دارد و توسط زمانبند کار انجام می‌شود.

ب) ایجاد و تخصیص ساختارهای لازم به این کار و یا به عبارت دیگر هر فرآیند در سیستم بوسیله بلوک کنترلی فرآیند (PCB) مشخص می‌شود.

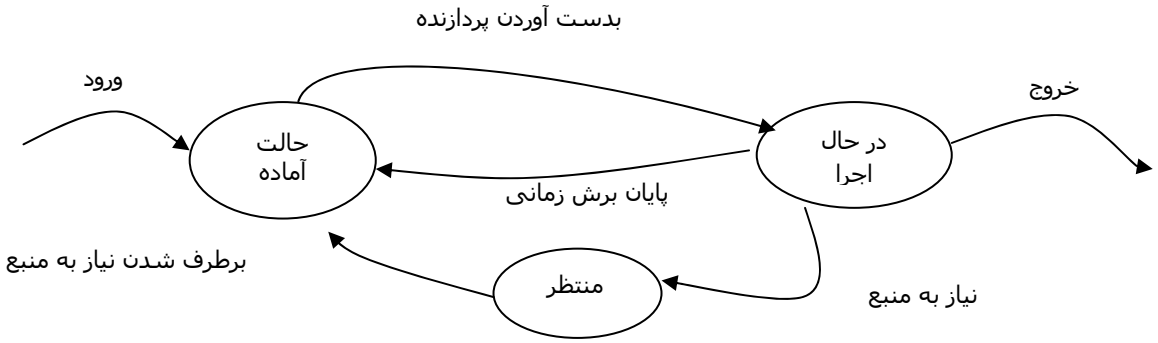
ج) قرار گرفتن در صف فرآیندهای آماده، بدین ترتیب هر کار پس از بدست آوردن PCB به فرآیند تبدیل شده و باید در نوبت اجرا یا صف فرآیندهای آماده (Ready) قرار بگیرد تا مراحل مربوط به زمانبندی پردازنده CPU Scheduling یا (زمانبندی کوتاه مدت) (Short – term Scheduling) در موردش انجام شود.

حالت‌های مختلف یک فرآیند (Process States)

1- حالت آماده (Ready State) – حالتی که فرآیند همه‌ی منابع لازم را در اختیار دارد و با بدست آوردن پردازنده، اجرای آن شروع می‌شود. هر فرآیند پس از ایجاد شدن یا بدست آوردن منابع مورد نیاز و یا پایان برش زمانی در این حالت قرار می‌گیرد.
2- حالت منتظر یا مسدود (Waiting – Blocked) – حالتی است که پردازنده در آن منتظر بدست آوردن یک منبع از سیستم می‌باشد. در این حالت اگر فرآیند

پردازنده را بدست آ ورد هم قابل اجرا نیست. فرآیند از حالت اجرا، با نیاز به یک منبع در این حالت قرار می‌گیرد.

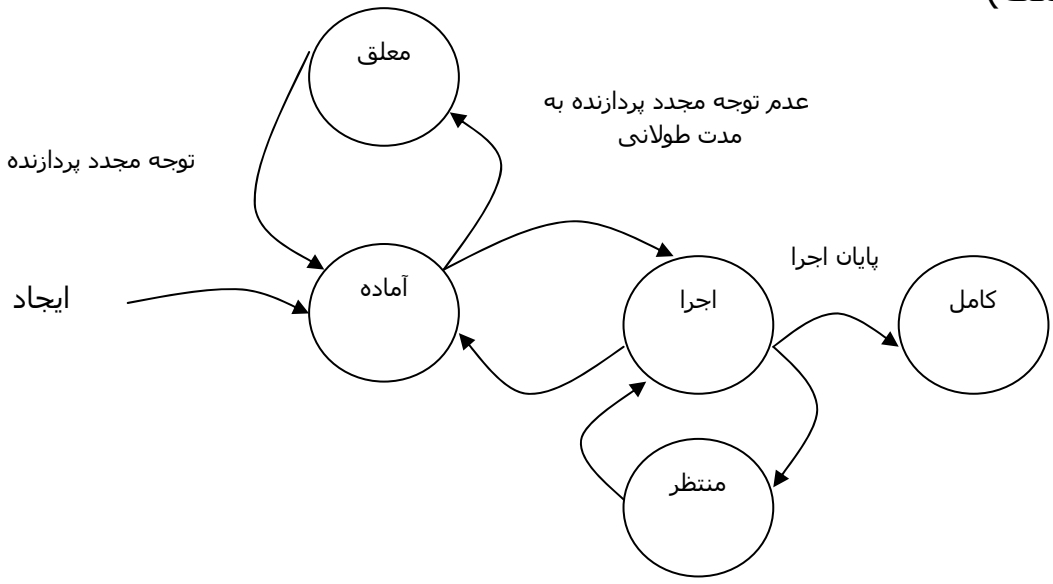
3- حالت اجرا (Running Executing) – در این حالت فرآیند همه‌ی منابع و پردازنده را در اختیار دارد.



حالاتی که در بالا توضیح داده شد حالات اصلی یک فرآیند بودند ولی یک فرآیند می‌تواند حالات دیگری را نیز داشته باشد.

4- حالت معلق Suspeud State) – حالتی است که یک فرآیند برای مدت طولانی پردازنده را بدست نیاورده است و حتما برای مدت زیادی نیز در این حالت می‌ماند (انتظار نامحدود)

5- حالت کامل – هنگامی که برآیند به اندازه کافی وقت پردازنده را بدست آورده و کارش خاتمه یافته است. (ولی هنوز از لیست فرآیندهای سیستم خارج نشده است)



فرآیند فرزند می‌تواند بطور همزمان با فرآیند پدر در حال اجرا باشد، و یا فرآیند پدر تا پایان اجرای فرآیند فرزند در حالت انتظار، باقی بماند و پس از آن اجرایش ادامه یابد. فرآیند فرزند می‌تواند بطور مستقل از منابع سیستم استفاده کند یا فقط از منابع در اختیار فرآیند پدر استفاده کند.

خاتمه‌ی فرآیند

هنگامی که کار یک فرآیند به پایان برسد و دیگر نیازی به آن نباشد و یا از منابع سیستم بیش از حد استفاده کند و یا پدر یک فرآیند در حال ختم شدن باشد، خاتمه یافته و از مجموعه فرآیندهای سیستم خارج می‌شود.

زمانی که یک فرآیند پدر خاتمه می‌یابد می‌بایشت تمامی فرآیندهای فرزند نیز خاتمه یابد یعنی خاتمه‌ی فرآیند پدر مشروط به خاتمه یافتن فرآیندهای فرزند است. که به آن ختم پی در پی Cascaded termination گفته می‌شود.

... خاتمه‌ی فزند فرزند → خاتمه‌ی فرزند → خاتمه‌ی فرآیند پدر

خاتمه‌ی فرآیند پدر → خاتمه‌ی فرزند →خاتمه‌ی فرزند فرزند → ...

رابطه‌ی بین فرآیندها

فرآیندها از جهت وابستگی به یکدیگر نیز در دو دسته قرار می‌گیرند:

الف) فرآیندهای مستقل – هر فرآیندی که داده‌ای را با دیگر فرآیندها به اشتراک نگذارد مستقل است و یا به عبارت دیگر یک فرآیند مستقل دارای شرایط زیر است:

1- حالت آن به اشتراک گذارده نشده باشد.
2- نتیجه اجرای آن کاملا مشخص باشد.
3- نتیجه‌ی اجرا در فعالیت مختلف یکسان باشد.
4- توقف و شروع آن بدون تاثیر بر سایر فرآیندها امکان‌پذیر باشد.

ب) فرآیندهای همکاری کننده – یک فرآیند همکار دیگر فرآیندهاست در صورتی که یکی از شرایط زیر در مورد آن صادق باشد:

1- حالت آن به اشتراک گذارده شود.

2- نتیجه اجرا بدلیل وابستگی به سایر فرآیندها کاملا مشخص نباشد.

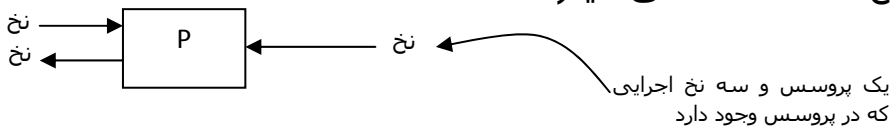
1- اجرا برای ورودی یکسان در فعالیت مختلف، یکسان نباشد.

رشته‌ها یا نخ‌ها (Threads)

یک فرآیند معمولی (طبق آن چیزی که گفته شد) فرآیند سنگین وزن است. ولی رشته‌ها به عنوان یک واحد پایه در ریزپردازنده‌ها سبک وزن هستند. این مفهوم مفهومی است که در سیستم عامل‌های جدید وجود دارد.

هر پروسس به دو بخش قابل تقسیم است:
- منابع در تملک پروسس
- عوامل اجرایی مانند Program Counter

درواقع جهت انجام خیلی از درخواست‌های کاربران به جای اینکه پروسس‌های جدید ایجاد کنیم از نخ‌ها استفاده می‌کنیم.



حسن روش‌های چند نخی – اگر یک نخ بلاک شود تمام پروسس از بین نمی‌رود آن نخ مورد نظر از بین می‌رود و گاهی اوقات یک مشکل را با یک پروسس نمی‌توانیم حل کنیم ولی با چند نخ از یک پروسس می‌توانیم مشکل خود را حل کنیم.

اگر یک نخ فایلی را باز کند تمامی نخ‌ها می‌توانند از آن فایل استفاده کنند و منبع که برای یک پروسس وجود دارد برای تمامی نخ‌های آن پروسس نیز وجود دارد.

کوچکترین واحد اجرایی هر پروسس نخ (Thread) می‌باشد

وضعیت‌های نخ Thread states
- آماده Ready
- اجرا Running
- مسدود Blocked

تمام نخ‌های یک پروسس در حالت و منابع آن پروسس شریک هستند. آن‌ها در یک فضای آدرس هستند و به داده‌های یکسانی دسترسی دارند. هنگامی که یک نخ یک عضو داده در حافظه را تغییر دهد نخ‌های دیگر تغییرات را می‌بینند و اگر یک نخ فایلی را باز کند نخ‌های دیگر می‌توانند از آن فایل باز بخوانند یا در آن بنویسند.

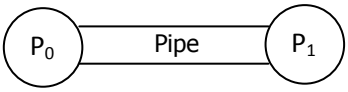
مزایای ایجاد نخ
- ایجاد یک نخ سریع می‌باشد.
- پایان دادن به نخ سریع است
- تعویض دو نخ در یک فرآیند سریع است. (سوئیچ کردن از یک نخ به نخ دیگر)

Job – در سیستم‌های قدیم به پروسس Job گفته می‌شد. Job یا فعالیت زمانی معنی می‌دهد که یک سیستم بصورت Batch کار می‌کند و off line است. ولی در سیستم‌های Interactive پروسس گفته می‌شود.

فایل یا پرونده – فایل مفهومی کلیدی است. فایل‌ها ساختارهای اطلاعاتی هستند که می‌توانند اطلاعات را بصورت ماندگار بر روی رسانه‌های ذخیره‌سازی نگه‌داری کنند. فایل یک مفهوم است.

Pipe یا لوله – یک شبه فایل مرتب است که برای اتصال بین دو پروسس به یکدیگر مورد استفاده قرار می‌گیرد. برای نگه‌داری اطلاعات نیست.

ویژگی مهم Pipe ترتیبی بودن آن است.



یعنی اطلاعات به همان ترتیبی که وارد Pipe می‌شوند به همان ترتیب هم خارج می‌شوند.

System call یا فراخوان سیستمی – هر درخواستی را که کاربران از سیستم عامل داشته باشند بازکردن، بستن، خواندن از فایل.

Shell یا پوسته – یکسری برنامه‌های سیستمی هستند یا قسمت‌هایی هستند که System callها را بین سیستم عامل و پروسس‌ها یا Userها حمل می‌کنند یا انتقال می‌دهند Editor, Command Interpreter.

فصل دوم

برای نصب لینوکس حدودا 11 گیگابایت فضا نیاز است. برای تهیه فضای مورد نیاز بصورت زیر عمل می‌شود:

My computer → Manage → Disk management
کلیک

لینوکس ؟؟ را برای نصب از برنامه Anaconda استفاده می‌نماید. این برنامه برای نصب آسان لینوکس طراحی شده و در هر قسمت از نصب راهنمایی درباره‌ی بسته‌های نصب شده را نشان می‌دهد.

بیشتر سیستم‌های برپایه‌ی اینتل با لینوکس سازگار هستند.

برای نصب از طریق CD ابتدا باید سیستم را برای راه‌اندازی از طریق CD تنظیم کرد.

Del → Setup → Advanced Bios Features → Firs Boot Device

در صفحه ابتدایی نصب گزینه‌های زیر وجود دارد:

Install or upgrade an exiting system

Install or upgrade an exiting system (text Model)

Rescue installed system

Boot from local drive

صفحه‌ی ابتدایی مربوط به قسمت CD

بعد از این صفحه Anaconda اجرا شده و روال نصب را در دست می‌گیرد.

Personal – Desktop – برای کاربران جدید مناسب است. این نوع نصب امکان بوت دوگانه را فراهم کرده و سیستم را برای بوت دوگانه پیکربندی می‌کند. برنامه‌ای بنام Grub امکان انتخاب و استفاده از ویندوز لینوکس را فراهم می‌کند.

Work station – بر پایه‌ی Personal Desktop می‌باشد. و برنامه‌های لازم برای توسعه‌ی نرم افزارها و مدیریت سیستم را دارا می‌باشد.

Server – نصب سرور برای سیستم‌هایی که میزبان یک وب سایت یا سایر سرویس‌ها هستند مناسب است. در این نوع نصب محیط‌های گرافیکی نصب نمی‌شوند بنابراین برای محیط‌های کاری مناسب نیست.

Custom – در این نوع تمامی جزئیات نصب را می‌توانید کنترل کرده و لینوکس را براساس نیازهای خود نصب کنید.

توجه – در نصب Server تمامی داده‌های روی هارد اعم از پارتیشن‌های ویندوز و غیر ویندوز را از بین می‌رود.

Remove all partition on selected drives create defult layout

Remove linux partitions on selected drives and create defult layout

Use free space on selected drives and create defult layout

Create custom layout

ایجاد پارتیشن‌ها بصورت دستی

Swap – در سیستم عامل لینوکس این پارتیشن بعنوان حافظه‌ی مجازی سیستم استفاده می‌شود زمانی که فضای Ram در هنگام اجرای برنامه کم باشد Swap

استفاده می‌شود. این پارتیشن برای کاربران قابل دسترسی نیست. فضای آن به اندازه‌ی دو برابر Ram است. (1042)

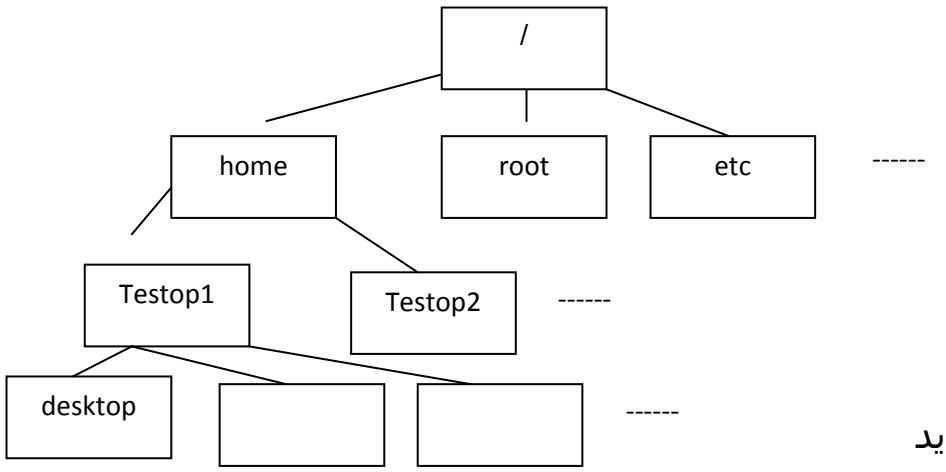
Boot – این پارتیشن برای نگه‌داری کردن لینوکس بکار می‌رود و در درایرکتوری boot/ نصب می‌گردد. حداقل اندازه‌ی این پارتیشن 100 مگا بایت باشد. بیشتر از این لازم نیست.

ریشه یا/ - این پارتیشن برای نگه‌داری سیستم عامل استفاده می‌شود و در دایرکتوری/ نصب می‌گردد. اندازه‌اش براساس تعداد بسته‌های نرم افزاری تعیین می‌شود.

Var - ایجاد این پارتیشن اختیاری است. این پارتیشن برای نگه‌داری اطلاعات متغیر همانند فایل‌های ثبت وقایع و غیره بکار می‌رود و باید فضایی حدود 3 گیگا بایت داشته باشد

Mount point ← Swap ندارد زیرا توسط لینکوس نصب نمی‌شود.

ریشه ← 3 گیگا بایت.



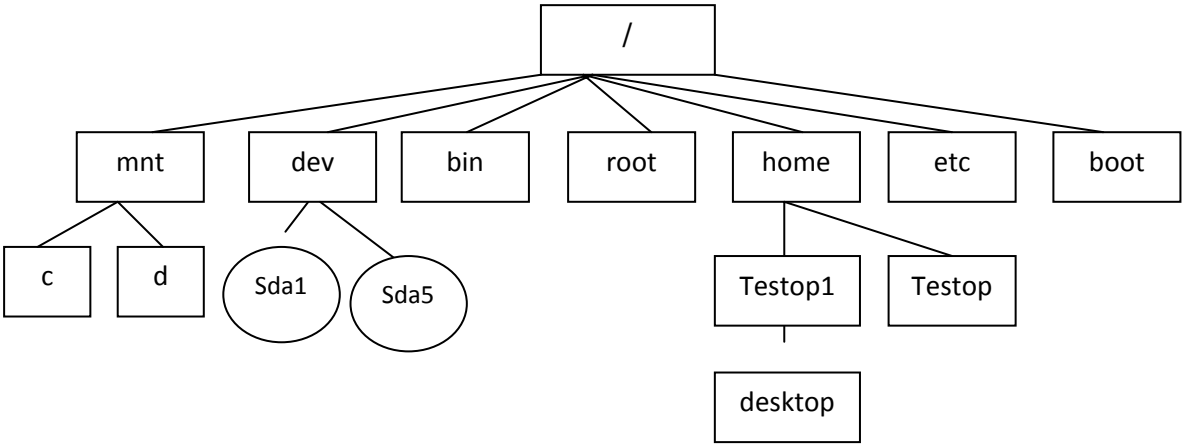
ایجاد کادر جدید

Computer → File system → Home → Testop 1

System → Administration→ User and Groups

سپس رمز عبور کاربر ریشه را وارد می‌کنیم.

پنجره‌ی User manager با Add user برای User جدید می‌باشد.



فایل‌های پیکربندی سیستم در ETC قرار دارد.

Grub یک فایل پیکربندی دارد با نام grub. Conf که در فایل etc می‌باشد. با کاربر root را وارد سیستم می‌شویم.

Computer → File system → Etc → Grub. conf

این برنامه باز می‌شود فایل پیکربندی برنامه‌ی Grub می‌باشد که می‌گوید Linux و ویندوز در کدام درایو و سکتور قرار دارد. تنظیمات grub، پیش فرض time out, grub برنامه‌ی grub چقدر باشد. در این فایل می‌توان سیستم عامل Defult را عوض کرد. می‌توان time out تغییرا داد.

برای تغییر سیستم عامل اگر Defult 0 باشد سیستم عامل پیش فرض Linux است و اگر 1 باشد سیستم عامل پیش فرض ویندوز است. برنامه‌ی grub در داخل پارتیشن بوت، boot است.

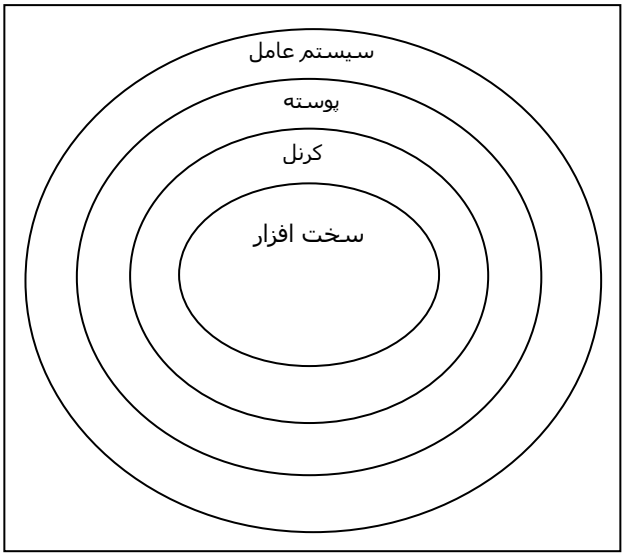
Dev ← اطلاعات سخت افزاری کامپیوتر را شامل می‌شود.

Application → System tools → Terminal

Su ↵

Password = 1 .. 6 ↵ برای ورود به کاربر ریشه

فصل سوم – دستورات پوسته



پوسته یا Shell برنامه یا واسطی است که بین کاربر سیستم عامل قرار می‌گیرد دستورات را از کاربر دریافت کرده و به سیستم عامل می‌دهد. بعد از اجرای دستورات نتایج را به کاربر برمی‌گرداند.

انواع Shell: Tcsh، Zsh، Pdks و Korn و ...

مهمترین Shell، Bash می‌باشد. انواع لینوکس دارای Shell bash می‌باشند.

برای دسترسی به Shell

Gnome → Application → System tools→ terminal

KDE → koonsole →Terminal →

KDE → Koonsole

Alt + Ctrl + fl → وارد محیط متنی می‌شوید

Alt + Ctrl + f7 یا f6 → برای خروج از محیط متنی

مدیریت دایرکتوری‌ها

mkdir نام دایرکتوری

mkdir doca

تغییر مسیر یا حرکت بین دایرکتوری‌ها

cd نام دایرکتوری

cd doca

بیرون آمدن از یک دایرکتوری

cd .. → UP

cd - → back

cd هر جا باشید شما را به دایرکتوری کاربر می‌رساند

cd / هر جا باشید شما را به دایرکتوری ریشه می‌رساند

حذف دایرکتوری‌ها

حذف دایرکتوری در صورتی که خالی باشد.

rmdir نام دایرکتوری

rmdir doca →

حذف دایرکتوری در هر صورتی چه خالی چه پر

rm – r نام دایرکتوری
rm – r doca

لیست فایل‌ها و دایرکتوری‌های داخل این شاخه

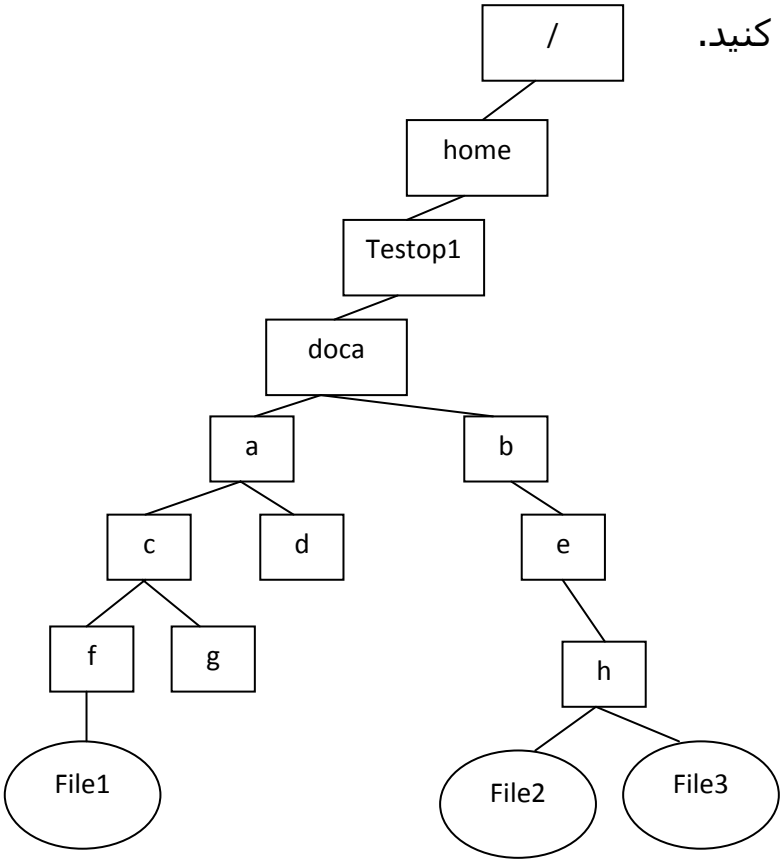
ls نام دایرکتوری

ls doca

ls → لیست فایل‌ها و دایرکتوری‌های جاری را نشان می‌دهد.

دستور tree برای مشاهده نام دایرکتوری tree ساختار درخت می‌باشد.

مثال: ساختار درختی روبه را ایجاد کنید.



مدیریت فایل‌ها

touch → ایجاد فایل
touch file

جهت ویرایش فایل‌ها از این دستور استفاده می‌شود. vi نام فایل ←

vi file1

a یا i فشار دهید ← ↳ i یا aکلید

Welcome to os lob

Esc ↲

: wq

Cat نام فایل ← نمایش محتوای فایل

نمایش محتویات یک فایل بصورت صفحه به صفحه
- More نام فایل
- Less نام فایل

sort نام فایل ← محتویات فایل بصورت مرتب شده نمایش می‌دهد.

دو دستور tail و head برای دیدن محتویات فایل است با این تفاوت n سطر اول را و n سطر آخر یک فایل را برای ما نمایش می‌دهد.

بصورت پیش فرض 10 خط
- نام فایل عدد tail
- نام فایل عدد head

اول و 10 خط آخر را نمایش می‌دهند.

head -15 نام فایل

tail -7 نام فایل

دستور کپی

cp مسیر فایل مقصد مسیر فایل مبدا

دستور انتقال

mv مسیر فایل مقصد مسیر فایل مبدا

cp – r مسیر دایرکتوری مقصد مسیر دایرکتوری مبدا

mv -r مسیر دایرکتوری مقصد مسیر دایرکتوری مبدا

file دستور file نوع فایل را مشخص می‌کند. → نام فایل

du → نام فایل/ نام دایرکتوری

حجم فایل یا حجم دایرکتوری را مشخص می‌کند.

یک فایل خالی 4 کیلو بایت و یک دایرکتوری خالی 8 کیلو بایت فضا اشغال می‌کند.
برای اینکه حجم را بطور خلاصه ببینید می‌توانید از S - استفاده کنید.

du – s نام فایل/ نام دایرکتوری

تنها حجم خود فایل یا دایرکتوری را نشان می‌دهد

man راهنمای دستور را بطور کامل نمایش می‌دهد → نام دستور

man touch

man bash

info نام دستور

راهنمای دستور را با استفاده از help - - - نام دستور help می‌توان مشاهده کرد.

در صفحات مربوط راهنما با b یا page up می‌توان بالا رفت و با استفاده از ؟؟ یا جهت یاب می‌توان سطر به سط به سمت پایین حرکت کرد. برای خروج از ctrl + z یا q استفاده می‌شود.

power off → خاموش کردن کامپیوتر

reboot → راه اندازی مجدد

با دو بار زدن کلید tab همه‌ی دستورات Shell نمایش داده خواهد شد.

برای خروج از Shell یا پوسته → Ctrl + D یا exit

لیست دستوراتی که تا کنون اجرا شده history ↵

بطور پیش فرض 500 دستور آخر

جهت خارج شدن از پوسته → exit یا ctrl + D

مسیر دایرکتوری جاری یا دایرکتوری که در آن کار می‌کنید را نمایش می‌دهد.

pwd → print working directory

date – → تاریخ و زمان

cal → تقویم ماه و روز جاری

cal → تقویم سال مورد نظر

توسعه‌ی نام فایل‌ها

* ← از صفر تا بی‌نهایت

؟ ← یک کاراکتر

[ ] ← به جای دو یا چند کاراکتر

rm doc*

به جای چند حرف (همه‌ی فایل‌هایی که با doc شروع می‌شوند)

به جای یک کاراکتر

rm doc

rm doc [a2] ↵

تمام فایل هایی که با doc شروع می‌شوند

و حروف چهارم آنها یا a یا 2 است.

{ } ← برای انتخاب مجموعه‌ای از فایل‌ها

touch doc {1, 2, a, b, ument, tor}

بازه‌ها

rm doc a

→ rm doc [a-h]

rm doc h

rm doc 1

→ rm doc [1-9]

rm doc 9

rm doc 1

→ rm doc {1..10}

Rm doc 10

حاصل ضرب دکارتی

{a , b} {1 , 2} → a1 , a2 , b1 , b2

touch {test , doc} {1 , 2}

Test 1

Test 2

Doc 1

Doc 2

rm doc [32] [27] → این روش غلط است

rm doc {32 , 57} → این روش درست است

مثال – فایل‌هایی را که با s شروع شده و به a ختم می‌شوند را حذف کنید.

rm S * a

مثال – فایل‌هایی را که با نام s شروع شده و شامل حرف a باشند را حذف کنید.

rm S*a*

مثال – فایل‌هایی را که 4 کاراکتری بوده و با s شروع شده و به a ختم می‌شوند را حذف کنید.

rm S??a

مثال – فایل‌هایی را حذف کنید که 4 کاراکتری بوده و با Sa شروع شده و به a یا b ختم شده‌اند.

rm Sa ? [ab]

مثال – فایل‌هایی را حذف کنید که 4 کاراکتری بوده و با Sa شروع شده کاراکتر سوم آنها a یا b باشند و کاراکتر چهارم آنها 2 یا 1 باشد.

rm Sa[ab][12]

مثال – فایل‌هایی را حذف کنید که با Sa شروع شده و کاراکتر سوم آنها a و b یا c باشد.

rm Sa [abc] *

مثال – فایل‌هایی را حذف کنید که با S شروع شده و به یک رقم ختم می‌شوند.

rm S* [0-9]

مثال – فایل‌هایی را به نام ایام هفته ایجاد نمائید.

touch {sun , mon , … } day

نام‌های مستعار

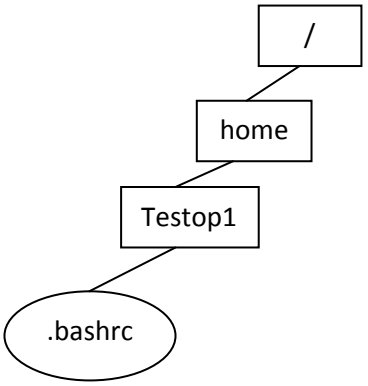
می‌توان برای هر یک از دستورات نام مستعار درنظر گرفت و بجای استفاده از نام دستورات از نام مستعار آنها استفاده کرد.

alias نام مستعار = نام دستور

alias md = mkdir (md نام دایرکتوری)

بعد از خارج شدن از Shell نام مستعار دیگر شناخته شده نیست. برای دائمی شدن این کار را در فایلی به نام bashrc. انجام می‌دهیم.

.bashrc. فایل پنهان



جهت دیدن فایل‌های پنهان دستور زیر را به کار می‌بریم:

Ls فاصله - a

touch. → نام فایل .

برای ایجاد فایل مخفی یا پنهان کردن فایل از نقطه قبل از نام فایل استفاده می‌شود

معمولا در بخش پیکربندی برنامه‌ها از فایل‌های پنهان استفاده می‌شود.

- ابتدا به دایرکتوری کاربر بروید. cd

vi .bashrc → -

(E)dit -
کلید a یا i برای شروع ویرایش

---
alias md = 'mkdir'

(داخل این فایل یک گروه از دستورات را نوشته با اجرای این فایل خط به خط دستورات نوشته شده اجرا می‌شود)

Esc ↵
: wq

با اجرای هر باره‌ی Shell، bashrc اجرا می‌شود پس با نوشتن اسم دستور مستعار در bashrc ما همیشه دستور مستعار را خواهیم داشت.

دستورات داخل فایل bashrc را داخل خود load کنید این دستورات برای shell شما شناخته خواهد شد.

هدایت (ارسال) خروجی به فایل:

man touch > test

cat test

> → با یک علامت بزرگتر محتویات قبلی از بین نمی‌رود.

>> → با دو علامت بزرگتر خروجی دستورات به محتویات قبلی اضافه می‌شود

Pipe یا لوله ¦

خروجی یک دستور را به عنوان ورودی برای یک دستور دیگر ارسال خواهد کرد.
ساختار درختی را بصورت صفحه به صفحه نمایش می‌دهد.

tree ¦ more →

cat ¦ test more

cat test ¦ sort

جستجو در فایل‌ها

grep کلمه یا عبارت مورد جستجو لیست فایل‌ها

grep ali test doc

grep ‘ the book’ test doc

grep ali *

فایل doc

ali is a boy

alireza is a boy

hasan is a boy

ali is a student

It’s invalid

Your name is amirali

grep ali doc

grep – c ali doc تعداد این کلمه (ali) در فایل doc را نشان می‌دهد.

grep – n ali doc → شماره سطرها را نمایش می‌دهد.

grep –w ali doc →

آنجاهایی که کلمه‌ی ali بدون پیشوند یا پسوند است را نشان می‌دهد.

fgrep " 1 عبارت

2 عبارت

3 عبارت

⋮

n عبارت"

fgrep “the book

ali " test doc

لینک‌ها

امکاناتی هستند برای دسترسی آسان‌تر یا سریع‌تر به فایل‌ها

لینک سمبولیک – محتویات فایل موجود نیست بلکه فقط لینکی از فایل اصلی را دارند اگر فایل اصلی حذف شود لینک سمبولیک دیگر کارایی ندارد.

لینک سخت – لینک‌های سخت همواره یک کپی از محتویات فایل اصلی را دارند بطوری که اگر فایل اصلی حذف شود اطلاعات داخل لینک موجود می‌باشد و قابل استفاده است. (در صورت حذف فایل لینک‌های سخت باز قابل استفاده هستند)

عیب لینک‌های سخت – به اندازه‌ی خود فایل اصلی فضا اشغال می‌کنند.

مزیت لینک‌های سمبولیک – فضای کمی اشغال می‌کنند به اندازه‌ی فایل خالی فضا اشغال می‌کنند.

ln نام فایل اصلی نام لینک سخت

ln -s نام فایل اصلی نام لینک سمبولیک

فشرده سازی فایل‌ها

gzip گزینه ها نام فایل

هنگامی که یک فایل فشرده می‌شود پسوند آن به gz تغییر می‌کند.

gzip test

برای unzip کردن یا خارج کردن یک فایل از حالت فشرده از دستور زیر استفاده می‌شود.

gunzip test .gz

gzip -v test

فایل را فشرده کرده و درصد فشرده سازی را نمایش می‌دهد.

gzip -l test .gz

یک خط اطلاعات در مورد فایل فشرده شده نمایش می‌دهد.

gzip -d test. Gz → برای unzip کردن

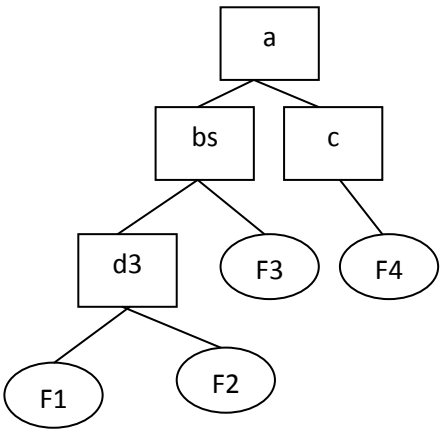
gzip -7 test → (درصد فشرده سازی بیشتر و سرعت فشرده سازی کمتر)
(درصد فشرده سازی را نشان می‌دهد).

gzip – r (a) → نام دایرکتوری

(r) ← برای فشرده سازی دایرکتوری‌ها

gunzip –r a → دایرکتوری را از حالت فشرده خارج می‌کند.

gzip -h → help برای فشرده سازی



نکته: دایرکتوری فشرده نمی‌شود بلکه فایل‌های موجود در آن دایرکتوری فشرده می‌شود.

bzip2 نام فایل گزینه‌ها

bzip2 test → پسوند فایل bz2. می‌شود

bunzip2 نام فایل فشرده شده

bunzip2 test. bz2

bzip2 -k test

-k(keep) خود فایل را فشرده نمی‌کند بلکه کپی فایل را فشرده کرده و نماش می‌دهد.

-f (force)

bunzip z -f test . bz2

-v(verison)

bzip2 -v test

bzip2 -l test

LL یا L- Ls اطلاعات کامل فایل را نمایش می‌دهد.

LL مخفف L – LS است.

– r – r × rw (نوع فایل و مجوزهای دسترسی به فایل)

3 کاراکتر اول مجوز دسترسی به فایل توسط مالک را نشان می‌دهد.

3 کاراکتر دوم مجوز دسترسی به فایل توسط گروه کاری را نشان می‌دهد.

سه کاراکتر سوم مجوز دسترسی به فایل توسط سایر کاربران را نشان می‌دهد.

- ← فایل عددی

d ← دایرکتوری

L ← لینک

r ← خواندن

w ← نوشتن

x ← اجرا کردن

chmod گزینه‌ها نام فایل

chmod u + x test

مجوز برای فایل test به مالک می‌دهد.

(u = user یا مالک)

ll test

chmod go – r test

(g = گروه کاری، O= سایر کاربران)

ll test

chmod u = rw test

مجوز خواندن را حذف می‌کند.

chmod a-rwx

همه‌ی مجوزها را از همه می‌گیرد یعنی دیگر هیچ کس مجوز دسترسی ندارد.

یکان دهگان صدگان 1 2 4

سایر گروه مالک x w r

مجوز دسترسی برای مالک فایل: u

مجوز دسترسی برای سایر کاربران: O

مجوز دسترسی برای گروه کاری: g

مجوزهای دسترسی برای مالک فایل، گروه و سایر کاربران: a

اضافه کردن مجوز: +

کم کردن یا حذف کردن مجوز: -

تعیین مجوز: =

chmod 764 test مالک فایل هر سه مجوز را دارد

گروه مجوز خواندن و نوشتن را دارد و سایر کاربران مجوز خواندن را دارند.

chomd 64 test. مجوزهای مالک از بین می‌رود.

chomd 4 test مجوزهای مالک و گروه از بین می‌رود و سایر کاربران باقی می‌ماند.

Home خانه کاربران، سایر کاربران که عادی محسوب می‌شوند در home ذخیره می‌شود.

Su (Supper user) ابر کاربر

Pass: 1-6

#

exit → برای خروج

دستور تغییر نام مالک فایل → نام فایل نام مالک جدید chown

chown ali a/b/doc

LL a/b/doc

---- ali ----

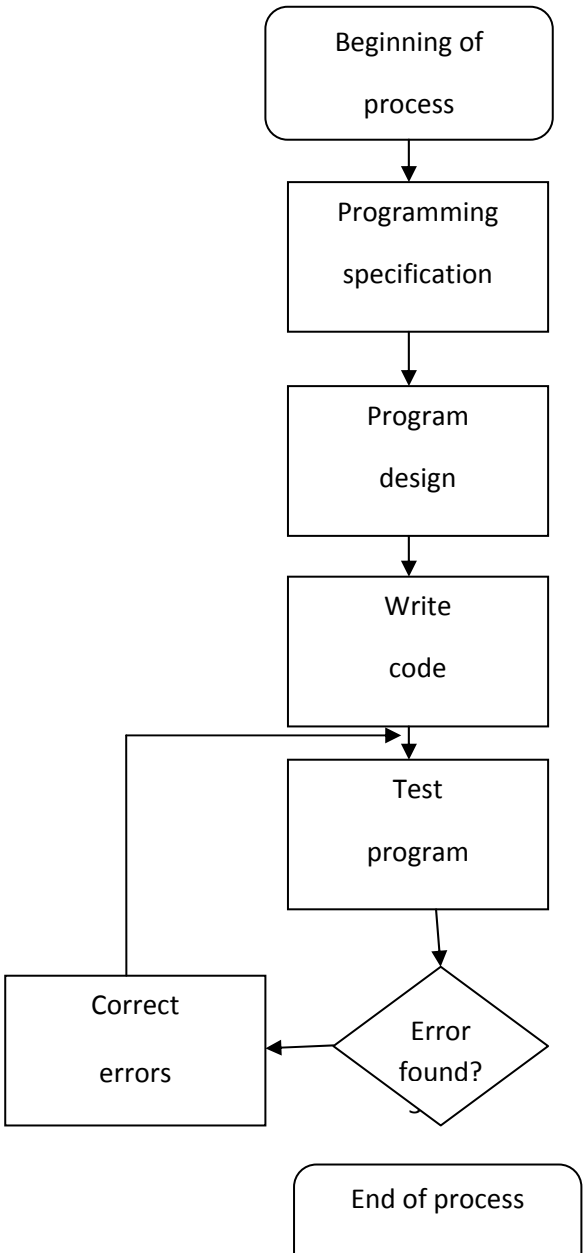
chgrp نام گروه جدید نام فایل

chgrp ali a/b/doc

---- ali ----

"shell programming"مقدمات برنامه‌نویسی پوسته

چرخه شکل گیری برنامه – پروسه‌ی شکل گیری یک برنامه کاربردی را چرخه شکل گیری برنامه می‌نامند چه این برنامه‌ها بصورت دست نویس‌هایی باشد چه یک برنامه به زبان سطح بالا.



1 - تعیین مشخصات برنامه شامل شرایطی که برنامه کابرد می‌یابد با آن‌ها تطبیق داده شود. 2- داده‌هایی را که برنامه به عنوان ورودی دریافت می‌کند 3- عملیاتی که قرار است بر روی داده‌ها صورت گیرد. 4- تعیین نوع خروجی برنامه نویسان می‌بایست قابلیت‌های مختلف فایل آرایش صفحه نمایش و الگوریتم‌های منطقی (روال‌ها) را خلق کنند.

برنامه‌های پوسته، فایل‌های متنی هستند که حاوی رشته‌هایی از دستورات لینوکس می‌باشند. یک برنامه نویس برنامه‌های پوسته را با یک ویرایشگر متن ایجاد می‌کند. برخلاف برنامه‌های نوشته شده به زبان‌های سطح بالا، برنامه‌های پوسته به یک کد زبان ماشینی تبدیل نمی‌شوند. به این دلیل که پوسته مانند یک مفسر عمل می‌کند.

همین که مفسر جمله‌ای را در فایل برنامه می‌خواند آن را فورا به دستورات قابل اجرا تبدیل کرده و آن‌ها را اجرا می‌کند. هیچ فایل exe (اجرایی) ایجاد نمی‌شود. زیرا مفسر جملات برنامه را در یک مرحله ترجمه و اجرا می‌کند و اگر یک اشتباه دستوری پیدا شود، برنامه متوقف می‌گردد.

پوسته برنامه نویسی – پوسته Bash به عنوان پوسته پیش فرض در مقام نسخه‌های لینوکس درنظر گرفته می‌شود.

کاربرد	نام پوسته‌ی اصلی و منشا	نام پوسته
امکانات برنامه نویسی قدرتمندی را هم چون متغیرهای پوسته ساختارهای کنترلی و عبارات منطقی/ تطبیقی به ارمغان می‌آورد	Korn, Bourn	Bash

عبارات پوسته از عملگرهایی نظیر عملگرهای زبان C بهره میگیرند	Cshell	Tsch
از بسیاری جهات شبیه پوسته Bash است اما فرم دستوری آن هم چون زبان C می‌باشد.	Korn	Zsh

پوسته Bash در مقایسه با سایر پوسته‌ها واسطه برنامه نویسی قدرتمندتری دارد.

متغیرها – متغیرها در واقع نامهای نمادینی هستند که به مقادیر ذخیره شده در حافظه اشاره می‌کنند.

- متغیرهای آرایشی
- متغیرهای محیطی Enviroment Variables
- متغیرهای پوسته Shell Variables

از متغیرهای آرایشی برای نگه‌داری و ذخیره اطلاعات مربوط به تنظیمات سیستم عامل استفاده می‌شود و آن‌ها را نباید تغییر داد.

از متغیرهای محیطی جهت مشخص کردن چیزهایی نظیر پوسته کاربردی، جایی که باید سیستم هامل به دنبال برنامه‌ها باشد و هم چنین مسیر فهرست ما در استفاده می‌شود سیستم عامل هنگام راه اندازی این متغیرها را می‌خواند.

متغیرهای پوسته – آن‌هایی هستند که در برنامه‌های پوسته یا در محل سطر فرمان ایجاد می‌شود. این متغیرها برای ذخیره موقت اطلاعات در برنامه‌های پوسته بسیار مفید هستند.

- Printernv برای نمایش متغیرهای محیطی

Printer nv ¦ More ← بصورت صفحه به صفحه نمایش می‌دهد.

کلید Space را تا آنجایی که خروجی به پایان برسد فشار می‌دهیم.

نام متغیر محیطی Printer nv ← تنها محتویات آن متغیر خاص را نمایش می‌دهد

PWD, USER NAME, SHELL, PATH, HOME, PS1

این‌ها نام برخی از متغیرهای محیطی و آرایشی هستند. برای اینکه بتوان متغیرهای پوسته را از متغیرهای محیطی و آرایشی تغییر داد متغیرهای محیطی و آرایشی با حروف بزرگ نمایش داده می‌شوند.

یک فایل دست نویس مقادیر اولیه متغیرهای محیطی را در فهرست مادر تعیین می‌کند.

نام	محتویات متغیر	تعریف شده بوسیله
HOME	مسیر نام فهرست مرجع مربوط به کاربر	سیستم
Long Name	شناسه کاربر	سیستم
TZ	منطقه زمانی به کار رفته توسط سیستم	سیستم
SHELL	مسیر نام برنامه برای پوسته مورد استفاده شما	قابل تعریف دوباره
PATH	لیست مسیر نام‌هایی که برای دستورات قابل اجرا جستجو می‌شوند	قابل تعریف دوباره

متغیرهای اولیه پوسته Bash

PS1	نشان اولیه پوسته	قابل تعریف دوباره
PS2	نشان ثانویه پوسته	قابل تعریف دوباره
PS3 , PS4	نشان‌های به کار رفته توسط دستورات set و select	قابل تعریف دوباره
IFS	علامت جدا کننده فیلدها	قابل تعریف دوباره

MAIL	نام فایل نامه که توسط برنامه Mail برای برنامه‌های رسیده مورد کاووش قرار می‌گیرد.	قابل تعریف دوباره
MAIL CHECKER	فاصله زمانی بررسی نامه‌های رسیده	قابل تعریف دوباره
MAIL PATH	لیست فایل‌های نامه که به mail برای پیغام‌های دریافتی بررسی می‌شود	قابل تعریف توسط کابر
TERM	نام پایانه	قابل تعریف توسط کابر
CD PATH	مسیر نامه‌هایی که دستور cd برای زیر فهرست‌ها کاووش می‌کند	قابل تعریف توسط کابر
EXINIT	دستورات راه‌اندازی برای ویرایشگر EX/Vi	قابل تعریف توسط کابر
PWD	فهرست جاری	قابل تعریف توسط کابر
OLDPWD	فهرست کاری قبلی که توسط دستور cd تعریف می‌شود.	قابل تعریف توسط کابر
REPLY	وقتی دستور read به تنهایی وارد می‌شود. شماره خط ورودی را باز می‌گرداند	قابل تعریف توسط کابر
UID	به هنگام راه اندازی پوسته ID کار برجای را برمی‌گرداند	قابل تعریف توسط کابر
EVID	به هنگام راه اندازی پوسته Effective ID کاربر جاری را باز می‌گرداند.	قابل تعریف توسط کابر
BASH	مسیر نام کامل Bash	قابل تعریف توسط کابر

BASHVERVION	نسخه Bash	قابل تعریف توسط کابر
SHELUL	هر بار که Bash راه اندازی شود، یکی به مقدار آن افزوده می‌گردد.	قابل تعریف توسط کابر
RANDOM	یک عدد تصادفی تولید می‌کند.	قابل تعریف توسط کابر
HISTSIZE	تعداد دستوراتی که سابقه آن‌ها ذخیره می‌گردد عدد پیش فرض 500 می‌باشد.	قابل تعریف توسط کابر
OPTARG	مقدار آخرین شناسه پردازش شده توسط دستور داخلی getopt	قابل تعریف توسط کابر
OPTFIND	اندیس شناسه بعدی پردازش شده توسط getopt	قابل تعریف توسط کابر
HOSTTYPE	نوع ماشینی که Bash بر آن اجرا شده است.	قابل تعریف توسط کابر
HOSTFILE	نام کامل میزبان سیستم	قابل تعریف توسط کابر
OSTYPE	مشخصات سیستم عاملی که Bash بر آن اجرا شده است.	قابل تعریف توسط کابر
ENV	نام فایل‌هایی که برای راه‌اندازی پوسته به کار می‌روند هم چون bashrc و tcshrc	قابل تعریف توسط کابر
HISTFILESIZE	حداکثر تعداد خطوط ذخیره شده در فایل سوابق	قابل تعریف توسط کابر
OPTERR	اگر مقدار آن عدد یک تعریف شود، Bash پیغام‌های خطی ایجاد شده توسط دستور getopt را نمایش می‌دهد.	قابل تعریف توسط کابر

قابل تعریف توسط کابر	دستوری را نگه می‌دارد که میبایست قبل از نمایش یک نشان اولیه اجرا گردد.	PROMPT COMMAND
قابل تعریف توسط کابر	ثانیه‌های مکث برای ورودی پس از نشان اولیه	TMOUT
قابل تعریف توسط کابر	ویرایشگر پیش فرض برای دستور FC	FCEDIT
قابل تعریف توسط کابر	فهرستی از پسوندهایی که به هنگام ایجاد نام فایل‌ها از آن‌ها صرف نظر می‌گردد.	FIGNORE

عملگرهای پوسته

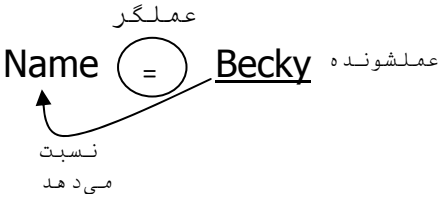
عملگرهای پوسته Bash به سه گروه تقسیم شده‌اند:

- عملگرهای تعریف و بازیابی
- عملگرهای عددی
- عملگرهای مجزا و تغییر مسیر

redirecting piping operators

تعریف و ارزیابی عملگرها

از علامت تساوی برای مقداری دهی یک متغیر استفاده می‌شود.



مثال – مقدار دهی و دیدن محتویات یک متغیر

Dog = Poodle↵

echo Dog↵ خروجی = Dog

echo $ Dog↵ خروجی = Poodle

مثال – تخصیص یک رشته از حروف به یک متغیر که حاوی حرف فاصله می‌باشد.

Memo = “Meeting will be at noon today” ↵

echo. $Memo خروجی = Meeting will be at noon today

مثال – نمایش حروف "و" که با متغیرها بکار می‌روند"

echo ‘$Home’ خروجی = $Home

مسیر فهرست جاری را بر صفحه نمایش = خروجی مشاهده می‌کنید.

echo “$Home”

مثال – نمایش کاربرد عملگر back quote فرمان date را اجرا کرده و خروجی آن را در داخل TODAY قرار می‌دهد.

TODAY = ‘date’

echo $TODAY خروجی دستور date را مشاهده می‌کنید

Echo – n $TODAY WriteLn

مثال – نمایش کاربرد دستور export

Cat > test script

echo → یک دست نویس که مقدار متغیر My – VAR را نمایش می‌دهد $MY – VAR

Ctrl + D ↵

Chmod ugo + X test ↵ این دست نویس را قابل اجرا می‌سازد.

script عدد 2 بر روی صفحه نمایش مشاهده می‌شود.

My – Var = 2 ↵

Echo $My – Var

در خروجی این دستور متغیر My – Var مشاهده نمی‌شود.

Printenv ¦ More ↵

اجرای دست نویس

./ test Script ↵

دست نویس یک خط را نمایش می‌دهد. چون که به متغیر پوسته دسترسی ندارد.

Export + My – Var

با این دستور این متغیر برای دست نویس قابل دسترس می‌گردد.

./testscript ↵ 2 : خروجی

Printenv¦ More

این بار متغیر My – Var دیده می‌شود.

نتیجه: دست نویس‌های پوسته نمی‌توانند بصورت خودکار به متغیرهایی که در سطر فرمان یا بوسیله دست نویس‌های دیگر ایجاد می‌شوند، دسترسی پیدا کنند. برای قابل دسترس ساختن یک متغیر برای یک دست نویس پوسته باید از دستور export استفاده شود.

مثال – نمایش محتویات متغیر مسیر

لیستی از فهرست‌ها «نام مسیرها با ء از هم جدا شده‌اند»

echo $PATH ↵

فهرست جاری را به مسیر جستجو اضافه می‌کند. به نموی که برنامه پوسته بتواند برنامه جدید را بیابد.

PATH = $PATH (:)

↵ (:)

به لیست اضافه شد. Echo $PATH

حالا برای اجرای دست نویس‌ها نیازی نیست که قبل از نامشان (./) را تایپ کنیم.

عملگرها

عملگرها	توضیحات
+ , -	مثبت و منفی
! , ~	نقیض منطقی
* , / , %	ضرب/تقسیم/باقی مانده
+ , -	جمع و تفریق
> , <	بزرگتر و کوچکتر
= , !=	تساوی و عدم تساوی

6 + 4 = 14

≠ 20

برای ذخیره مقادیر عددی در یک متغیر از جمله Let استفاده می‌شود.

Let X = 6 + 4 * 2

Let Y = X + 4

تمرین – استفاده از عملگرهای عددی

Let X = 10 + 2 * 7 ↵

echo $ X↵ خروجی = 24

Let Y = X + 2 * 4 ↵

echo $Y ↵ خروجی = 32

اعداد مبنای 8 با 0 آغاز می‌شوند.

اعداد مبنای 16 با X مشخص می‌گردند.

این خط سبب می‌شود که از دوباره نویسی در یک فایل جلوگیری شود.

Set –o noclobber ↵

Cat new – file > old – file

Cat = file exist ↵

Cat = new file > ! old – file

علامت ! باعث می‌شود که دوباره نویسی انتخاب شود.

عبارات عام

* , [, ?

تمام فایل‌هایی که پسوند C دارد را استخراج می‌کند. S * C

عبارات عام به نام حروف **Glob** گلاب نیز نامیده می‌شوند.

علامت سوال (?) – بطور دقیق با یک کاراکتر تطبیق پیدا می‌کند به غیر از ؟؟ و (.)

علامت (*) – با هیچ یا چند حرف تطبیق پیدا می‌کند.

[Chars] – با یک سری حروف تطبیق پیدا می‌کند. حروفی که در داخل کروشه تعریف می‌شود.

more chap [1-3] ↵

این دستور محتویات سه فایل Chap 1 را نمایش می‌دهد.

Chap 2

Chap 3

عملگرهای رابطه‌ای

- gt >

-ne ! = -Le <= -Lt <

-eq = -ge > =

عملگرهای منطقی

and or not

-a -o !

مثال – برنامه‌ای بنویسید که یک عدد را از ورودی گرفته و معکوسش را چاپ کند.

32 → 23

```
Read a
let  b1 = a % 10
echo  –n $b1
let  a = a / 10
let  b2 = a % 10
echo  –n  $b2
```

ساختارهای منطقی – منطق تصمیم‌گیری
- منطق ترتیبی
- منطق حلقه‌ای
- منطق موردی

منطق ترتیبی – نشان دهنده آن است که دستورات به همان ترتیبی اجرا می‌شوند که در برنامه آمده‌اند. این روال هنگامی متوقف می‌شود که یک دستور انشعابی مسیر اجرای دستورات را تغییر می‌دهد.

مثال:

```
Vi  Seqtotal
let a = 1
let b = 2
let c = 3
let  total = a + b + c
echo  $total
esc↵
: wq  ↵
sh   seqtotal ↵ خروجی = 6
```

منطق تصمیم گیری – این منطق برنامه شما را قادر می‌سازد تا یک یا چند جمله را تنها وقتی اجرا کند که شرایط خاص برقرار باشد. (برقرار بودن یک شرط) عبارت if ساختار کنترلی اولیه برای تصمیم‌گیری در این منطق را تشکیل می‌دهد.

حالت اول:

```
if        شرط
 then
    دستورات
fi
```

حالت دوم

```
if        شرط
  then
        دستورات 1
else
```

دستورات 2

fi

حالت سوم

if شرط

then

دستورات 1

else

if شروط 2

then

دستورات 2

else

دستورات n

fi 1

fi 2

fi n

مثال 1 – [] معادل دستور test می‌باشد.

```
read X
If   test   $X  -gt  5
یا
If   [ $X  –gt  5 ]
then
clear
```

```
echo  $x
echo  "hello"
fi
```

مثال 2

```
Read X
if   [ "$x"  = "Linux" ]
then
 clear
echo  $x
else
echo "error"
fi
```

مثال 3

```
read  x
if  [ "$x" = "a" ]
 then
 clear
ls  -a
else
if  [ "$x" =  "L" ]
  then
  clear
ls  –l
```

```
else
if  [ "$x" =  "R" ]
  then
 clear
ls  –r
else
if  [  "$x" =  "r" ]
  then
 clear
 ls – r
else
 clear
ls
fi
fi
fi
fi
```

تمرین معدل دانشجو

20 → good

17 – 19 → A

14 – 17 → B

10 – 14 → C

0 – 10 →D

در غیر اینصورت → error

```
echo  "please enter your average"
read a
if  [ $a =  20 ]
then
echo "good"
else
if  test  $a  –ge  17  –a  $a   -Le  19
then
echo "A"
else
if  test  $a  -ge  14  –a  $a  lt  17
then
echo "B"
else
if  test  $a  ge  0  –a  $a  lt  10
then
echo "D"
else
echo "error"
fi
fi
fi
fi
fi
```

دستور Case

case متغیر In

1 حالت) دستورات 1 ;;

2 حالت) دستورات 2 ;;

n حالت) دستورات n ;;

esac

مثال

```
read Color
case $color in
"blue") echo "Esteghlal" ;;
"red") echo "perspolis" ;;
"white") echo "melli" ;;
*) clear
echo "Error" ;;
esace
```

مثال

```
echo "please characters"
Read X
case $x in
"a") ls –a ;;
```

```
"L") ls  –a ;;
"R") ls  –l ;;
"r") ls  –r ;;
*) ls clear
echo  "error";;
esac
```

حلقه for

حالت اول

```
for  x  in 1   2   3   4
do
دستورات
echo  $x
done
```

مثال

```
for  name in  ali  reza  hasan
do
....
done
```

حالت دوم

seq 10 → اعداد 1 تا 10 را نشان می‌دهد

seq 3 20 → 2 پارامتر داریم اعداد 3 تا 20 را نشان می‌دهد

seq 3 2 20 → 3 تا پارامتر داریم 20 ... 9 7 5 3

```
for x in "Seq 10"
do
...
done
```

```
 for  x  in  "Seq 3 30"
        do
         ...
        done
```

مثال –

```
for  x  in  "Seq 20 -1  1"
do
...
done
```

حالت سوم

```
for ((i=1 ; k=10 ; i=i+1))
do
...
done
```

تمرین – برنامه‌ای بنویسید که n تا اسم از ورودی بگیرد و بصورت مرتب نشان دهد. ابتدا اطلاعات را داخل یک فایل ریخته سپس اطلاعات فایل را Sort کنید.

```
read  n
for (( i =a ; i<=n ; i=i+1))
do
read  name
echo  "$name" >> test
done
sort  test
```

فایل کمکی گرفته شده در انتهای برنامه آزاد می‌شود.

برای دومین بار اجرای برنامه محتویات قبلی را باز rm test نشان می‌دهد. برای حذف آن بهتر است در انتها برنامه فایل را حذف کنیم.

دستور While

```
While شرط
do
دستورات
done
```

دستور Until

```
until شرط
do
دستورات
done
```

مثال – عدد m رقمی را بگیرد و بصورت معکوس نمایش دهد.

```
Read a
While ( $a  – gt  0 )
do
let  b = a%10
echo  –n  $b
let  a =a/10
done
```

آرایه‌ها

```
a[1] = 5
a[2] = 3
Let  a[3] = a[1] + a[2]
echo  ${a[3]}


a[1 , 1] = 5
a[1 , 2] = 3
Let a[2 , 1] = a[1 ,1] + a[1 , 2]
echo  ${a[2 , 1]}


a[1] = ‘ali’
a[2] = ‘a’
```

تمرین – برنامه‌ای بنویسید که مقادیر داخل یک آرایه را مرتب کند.

```
Read n
for ((i = 1 ; I <= n ; i = i + 100
do
    read a[i]
done
for ((i = 1 ; I <= n ; i = i + 1))
do
for ((j = i + 1 ; j <= n ; j = j + 1))
do
if  [ ${a[i]  – gt ${a[i]} ]
then
let  temp = a[i]
let  a[i] = a[j]
let  a[j] = temp
fi
done
done
for ((i = 1 ; i <= n ; i = i + 1))
do
echo ${a[i]}
done
```

روش ایجاد توابع

نام تابع ()

{

دستورات

}

تابع‌ها را می‌توان در سه مکان مختلف ایجاد کرد.

1- در shell: تا جایی که از این محیط خارج نشویم معتبر است.

2- در داخل یک فایل ایجاد کنیم و بعد هر جا که نیاز است فراخوانی شود.

حالت اول

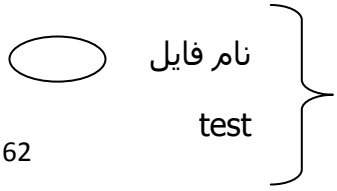
```
F( )
{
        Clear
            Echo "hello"
}
F1 ↵
```

حالت دوم

```
Vi test
F2 ( )

Clear
Date
}
```

اجرا کردن ←



1- load کردن فایل

2- f2 ↵

حالت سوم

```
Vi boshrc
F3 ( )
{
Clear
Ls
}
F1 ( )
{
Echo ‘hello’
}
F2 ( )
{
Clear
Date
}
Read X
If [ $X  –gt  S ]
Then
F1
Else
```

F2

fi