**Linux** – eshte system operativ me I perhapur per serveret ne internet. Ai eshte FREE si dhe Multi-user dhe Multi-tasking.

**Multi-User** – dmth qe resurset e sistemit operativ(rami,procesori…etj) ndahen ne menyre te drejte ndermjet shfrytezueseve.

**MultiTasking** – Shume procese ekzekutohen njekohesisht

Komponentet kryesore te Unix/LINUX jane:

1. KERNEL
2. SHELL
3. PROGRAMET

**GREP**- eshte komande me ane te te cilies mund te bejm kerkimin e shprehjeve ne kuader te nje fajlli tekstual.

**Process** – quhet programi ne ekzekutim dhe rrethina ne te cilen ai ekzekutohet.

Cdo process e ka numrin e vet identifikues te quajtur **PID** (process Id)

Preceset komunikojne permes sinjaleve. Nje process mund te nderprehet me komenden **KILL.**

**Root –** eshte administrator I sistemit ne Linux

Ekzistojne dy lloje te variableve: lokale dhe globale.

Variablat lokale- ekzistojne vetem ne shell

Variablat globale – ekzistojne edhe ne rrethinen e te gjitha programeve

Strings - jane vektor i char.

Ndryshe quhen edhe lidhje e shenjave qe perfundon me ‘\0’

**Pointeret -** jane variabla qe ruajne adresat memorike ten je variable tjeter.Ata mundesojne call-by reference.

Deklarimi i pointereve behet ne kete menyre: int \*myptr

Ja nje shembyll:

Int y = 5;

Int \*yptr;

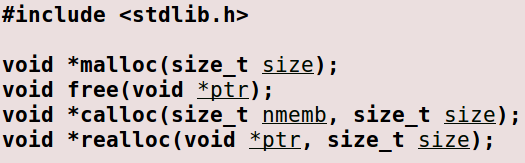
yptr= &y; -- inicializimi I pointerit

**Operatori &** - Përdorët për ta lexuar adresën memorike të një variable

Pointeret na ndihmojne ta perdorim HEAP MEMORY (alokimin dinamik te memories).

void \*malloc - rezervon memorien

void free – e liron memorin



**Serializimi** nënkutpon procesin e konvertimit të gjendjës së një objekti në një format të ruajtjes së të dhënave, p.sh. në disk, apo në rrjetë

**Deserializimi** nënkutpon procesin rikthimit të gjendjës së objektit të serializuar nga formati i ruajtjës së të dhënave, p.sh. nga disku, apo nga hostet tjera në rrjetë

Sfidat e serializimit jane:

* Byte Order (Endianness)
  + Renditja e byte-ave gjatë reprezentimit binar të numrave, para së gjithash int
  + Makina të ndryshme përdorin Endianness të ndryshëm
  + Problemi lind kur makina me Endianness jo kompatibël lexojnë shënimet e shkruara nga makina tjetër
* Endianness më të përhapur:
  + **Big-Endian:** 
    - motorola-format ndryshe njihet si ***network byte order***
    - ***Power-PC, Raspberry Pi, etc.***
  + **Little-Endian:** 
    - intel-format shumë i përhapur në domainin e PC-ve.
* Serializimi/Deserializimi arrihet përmes sys-calls:
  + ***hton*** gjatë shkruarjes/transmetimit të variablave int & Co. në disk/rrjetë
  + ***ntoh*** gjatë leximit/interpretimit të variablave int & Co. nga disku/rrjeta

**Proces** – quhet nje program ne ekzekutim

Procesi perbehet nga (ID,Variabla,Staku)

* **Multitasking** - Aftësia e një sistemi operativ për të ekzekutuar më shumë një program njekohësisht.
* ***Time-sharing***
  + Resorset lirohen në bazë kohore, ose në bazë të prioriteteve.
  + Në kete menyre një applikacioni nuk mund ti bllokoj resorset kur ato nevojiten nga applikacioni tjetër.
* Sjellja“ (behaviour) i një procesi karakterizohet me ane të listës së komandave të ekzekutuara të një procesi
* Një liste e tille quhet ***trace***

**Gjendjet e procesit –**

* *Ready*
  + Procesi është i gatshem për ekzekutim dhe pret për tu ekzekutuar nga CPU.
* *Blocked*
  + Procesi pret për një event (ngjarje) (p.sh. perfundimi i një operacioni I/O) – procesi nuk është në gjendje për tu ekzekutuar

*Running*

* + Procesi është në posedim të CPU dhe është në ekzekutim

**Cka eshte Swapping?**

* *Swapping* = shkarkimi i proceseve në një memorje sekundare (p.sh. Harddisku)
* Për realizim të kesaj në OS na nevojiten 2 „gjendje“ dhe dy „queue“ të reja
  + *Ready, suspend*

*Blocked, suspend*

* Numri unik i procesit – **PID** (proc. identifier) është Indeks në tabelen primare të proceseve (Primary Process Table)
* Identifikimi i shfrytezuesit (User identifier)
  + Shfrytezuesi të cili i takon procesi
  + Numri i procesit të cilin e ka gjeneruar procesin (Parent Process Identifier) – PPID
* Për ti mbrojtur shtrukturat e të dhenave të OS, ekzistojne dy mode të ekzekutimit
  + Modi i previligjuar (system mode, kernel mode, supervisor mode, control mode)  
    përdoret për qasje në regjistrat kontrollues, instruksione primitive të I/O, MM
  + Modi i shfrytezuesit (User mode)
* *Mode Switch* në mes të *User/Kernel Mode*
* *Process Switch* behët nderpreja e ekzekutimit të procesit për shkak të *interaptave*, *gabimeve* ose *Supervisor Call*
* Programet dallojne prej proceseve
* Proceset jane programe që ekzekutohen (në ekzekutim)
* Disa procese në menyre konkurente (të „njëkohësishme) mund ta ekzekutojnë të njejtin program
* Procesi *init* është procesi kryesor i cili është paraardhesi i të gjitha proceseve të systemit ose të shfrytezuesve.

**System Calls** - jane servise qe sistemi operativ ja jep programeve te shfrytezuesve ne disponim.

System calls mund te “thirren” (invoke) prej programeve te ndryshme

**Fork** - krijon një proces tjeter duke e kopjuar procesin ekzistues. Procesi i ri i krijuar quhet procesi femijë (child process). Procesi i cili ekzekuton fork quhet procesi prind (parent process). Procesi femije i trashigon prej procesit prind

* Fajllat e hapur (çasje e perbashket)
* Definimet e sinjaleve
* Vlerat momentale te variablave

Mirepo :

* Variablat jane lokale (nuk ndikohen nga njeri process)
* Sinjalet mund te perdoren poashtu ne menyre locale.

Procesi femij dhe prind i kane te njejtat variabla përveq variables pid.

**pid** **-** e dallon procesin prind prej atij femijë.

**pid** **-** i processit femij është 0 , ndersa i procesit prind është numer i plote pozitiv jo-zero.

**Exec** - transformim i procesit duke e mbuluar hapsiren memorike me ate të një programi tjeter. Exec perdoret për më i ekzekutuar programet tjera. Rezultati mund të kuptohet si nje proces tjeter, në të vertete procesi i vjeter shkrihet në të keshtu ekzekutohet nje program tjeter por i cili e ka të njejtin proces-id sikurse procesi që e therret.

**Wait** - përdoret për sinkronizim të proceseve, një process pret deri sa të perfundoj procesi femij (tjetër). Ndryshe mund te themi e „suspendon“ perkohësisht processin thirres deri sa procesi femij të terminoj (perfundoj).

* wait e merr një argument – status
* Status = NULL nese argumenti nuk meret parasysh
* Statusi e merr vleren e exit-statusit të procesit femij
* Kombinimi i *fork* dhe *wait* i dobishem kur deshirohet qe nga programi të ekzekutohet një program tjeter me anen e *exec*

**Exit** – shfrytezohet për me e perfundue (terminuar) procesin femije. Programi perfundon edhe kur ta arrij fundin ose kur ta arrij return komandon

**Atributet e proceseve**

Secili proces i ka disa atribute permes të cilave Sistemi operativ i kontrollon, i menagjon, e mirembane sigurine e file –sistemit.

Nje atribut është: pid Ai lexohet me kete komande: **pid=getpid()**

* Group-id – proceset që kan lidhje më nje pipe
* Sesion-id – poceset që kane lidhje më nje sesion
* Sesioni krijohet si të beni login-in në sistemin operativ si shfrytezues

**Sinjalet dhe procesimi i sinjaleve**

Programet e „medha“ të ndahen në grupe të programeve të „vogla“ - per shkak te memorjes jo te mjaftueshme.

Inter-Process Communication - IPC“ kommunikimi mes proceseve mund te behet permes:

* Signals
* Pipes dhe named pipes (FIFOs)
* Message queue
* Shared memory
* Sockets

Sinjalet shfrytezohen prej OS për të kontrolluar proceset në rast të gabimeve.

**Sinjalet perdoren për**

* ta nderprere ekzekutimine programeve
* në rast të kushteve „jo-normale“ (gabimeve)
* jo për transmetim të shenimeve në mes të dy proceseve
* **PIPES**- Kanal kommunikim në mes të procesve „*farefisnore*“.

**Vetite**:

* Nje-direksional
* „Stream“ i shenimeve
* Per transmetim bi-direksional (në të dy drejtimet) duhen dy pipes.

**Pipe** deklarohet permes nje vektori me dy elemente

int p[2];

* p[0] është „Skaji i leximit“
* p[1] është „Skaji i shkruarjes“

**dup2** perdoret per redirektim te pipes ose fajllave

**Dallimi ne mes te PIPES dhe Message Queues**

**Named Pipes**

* Të orientuara në streams
* Leximi behet në menyre sekuencuale
* Probleme me me shume Shkrues/Lexues

**Message Queues**

* Të orientuar në paketa
* Leximi selektiv
* Me shume shkrues se lexues – pa probleme

**SHARED MEMORY**

Shared memory (SHM) operacionet i lejojne dy ose me shume procese te “ndajne” nje segment te memorjes fizike. Mekanizem shume efikas per IPC shkruarje “direkte” ne memorje. Ajo ne vete permbane 4 operacione:

Krijimi, Kycja,Ckycja,Fshierja.

**Seksioni kritik (critical section)**

* Një proces gjendet në seksioni kritik kur ai i qaset resourseve (shenimeve) të perbashketa (shared memory).

**Mutual Exclusion**

* Në të njejten kohe vetem një proces guxon të gjindet në seksionin kritik
* Qasja në seksionin kritik duhet te behet në menyre të renditur
* Pengon qasjen e njekohësishme në resurse

Nje proces **ndodhet** në **seksionin kritik** kur ai i qaset sehnimeve të përbashkëta

* Hyrja në seksionin kritik duhet të behet në menyre të renditur (rregulluar)
* Asnjë proces nuk guxon të hyje në seksionin kritik nëse një proces tjetër gjindet në seksionin kritik

Disa lloje te zgjidhjeve jane:

**Zgjidhjet softverike**

* Nuk ka parashikime për funksionim korrekt
* Sinkronizimi përmes variablave globale
* Busy Waiting: pritja deri sa të plotësohet kushti duke pyetur non-stop

**Zgjidhjet e Sistemit Operativ**

* I ofrojne programerit funksione dhe struktura të shenimeve
* Semaforet ose mekanizma e tjere

**Semaforet**

* Mekanizem sinkronizues i cili e perjashton „busy waiting“ (e ofron sistemi operativ)
* Semaforat S: Integer-Variable, me të cilen dy operacion atomare kryhen, wait dhe signal (Dijkstra)
* Në vend të Busy Waiting: i bllokon proceset në Blocked Queues
* “Variabel e perbashket” S për sinkronizim
* Operacionet Wait(S) respektivisht P(S):
* Procesi guxon të hyje në seksionin kritik atehere kur SEMAFORI S>0, në të kunderten procesi bllokohet
* Nuk ka busy waiting, por bllokim te procesit prej sistemit operativ
* Procesi i qaset seksionit kritik dhe e inkrementon semaforin
* Operacionet Signal(S) respektivisht V(S)
* Procesi del prej seksionit kritik dhe e dekrementon semaforin

* Nese disa procese presin nje semafor (S) atehere ato do të vendosen në „Blocked Queue“ të semaforit S.
* Signal e largon nje proces prej „Blocked Queue“ dhe e vendose në „Ready Queue“

**Inicializimi: Semaforet duhet të inicializohen me nje vlere jo-negative**

* Semaforet binare : marrin vlerat 0 dhe 1
* Counting Semaphore : marrin vlera të ndryshme (jo negative) – varësisht prej implementimit

Me cdo semafor jane te shoqeruara keto variabla:

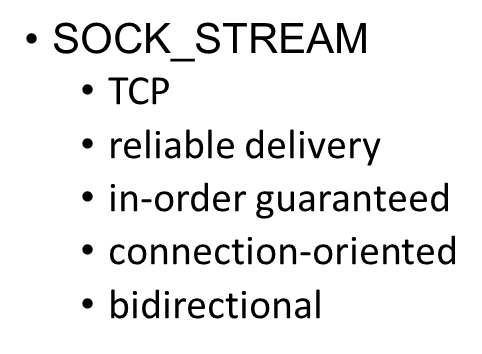
* semval – vlera e semaforit
* sempid – PID is procesit qe e ka semaforin
* semcnt – nr i proceseve qe jane duke pritur per vleren me te madhe se vlera momentale

**Cka eshte socket?**

Nje interfejs ne mes te aplikacionit dhe rrjetes (Ethernet)

* Aplikacioni e krijon socket
* Tipi i socket e dikton stilin e komunikimit
  + reliable vs. best effort
  + connection-oriented vs. connectionless

**Dy tipet esenciale te sockets**

****

