**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.5

*la Sisteme de operare*

**Au efectuat: Vlasitchi Stefan**

**st. gr. TI-216**

**A verificat: Colesnic Victor**

**asist. univ.**

**Chişinău - 2023**

**Lucrare de laborator nr. 6**

**Tema:** Gestionarea proceselor

Subiecte de pregătire către lucrarea de laborator 6: Explicaţi comenzile – kill, killall, pidof, pgrep, pkill, nice, renice, at, tail, sleep, cron, trap. Explicati notiunea de handler..

Noțiuni teoretice.

Comenzi pentru controlul proceselor. Prezentăm unele comenzi utile (descrierea detaliată a caracteristicilor și a sintaxei comenzilor - man comanda (sau info comanda)):

* kill – transmite un semnal procesului. Semnalul poate fi un număr sau un nume simbolic. În mod implicit (fără a specifica semnalul), se transmite semnalul de terminare a procesului. Identificarea procesului se petrece după PID. Utilizaţi comanda kill –l şi primiţi o listă a semnalelor de sistem disponibile în GNU/Linux.
* killall - funcționează în același mod ca și comanda kill, dar pentru identificarea procesului utilizează un nume simbolic în loc de PID;
* pidof – afişează ID-ul procesului a unui program în execuție;
* pgrep – determină PID-ul proceselor cu caracteristicile specificate (de exemplu, lansate de un anumit utilizator);
* pkill – trimite un semnal unui grup de procese cu caracteristicile specificate;
* nice – lansează procesul cu valoarea de prioritate specificată. Reducerea valorii (creșterea priorității execuției) poate fi inițiată numai de utilizatorul root;
* renice – modifică prioritate (valoarea) procesului lansat. Reducerea valorii (creșterea priorității execuției) poate fi inițiată numai de utilizatorul root;
* at – comandă introdusă de la tastatură, care urmează să fie executată ulterior;
* tail - vă permite să depistaţi și afişaţi liniile la sfârșitul fișierului;
* sleep – introduce o pauză în execuţia script-ului;
* cron – un daemon care permite execuţia comenzilor la un moment de timp.

cron rulează în background și verifică în permanență directoarele /etc/cron.daily, /etc/cron.hourly

/etc/cron.monthly și /etc/cron.weekly pentru a găsi script-urile adăugate. Comenzile sunt specificate în fișierul /etc/crontab (prin adăugarea de

linii în fișierul crontab sau folosind comanda crontab). Comenzile ce vor fi executate doar o singură dată sunt adăugate folosind at.

Pentru a deschide fișierul de configurare cron, se va executa comanda:

$ crontab -e

Structura unei linii constă din următoarele câmpuri:

Structura unei linii constă din următoarele câmpuri:

minute

luna

cale/către/c omandă

m h dom mon dow user comand

ora

zi-din-lună

zi-din- săptămână

utilizatorul

Unde:

* minute: 0 – 59;
* ora: 0 – 23;
* ziua-din-lună: 0 – 31;
* luna: 1 – 12 (12 == decembrie), sau *jan, feb, mar*,…;
* ziua-din-săptămână: 0 – 6 (0 == duminica), sau *sun, mon, tue, wed, thu, fri, sat;*
* cale/către/comandă: calea către script-ul (comanda) care trebuie să fie executat.

Exemple de utilizare.

Linia de mai jos execută script-ul backup.sh în fiecare zi a săptămânii la ora 2 dimineața:

0 2 \* \* \* /root/backup.sh

Linia următoare se execută în fiecare zi din două în două ore, începând cu ora 0:23, continuând cu 02:23, 04:23, etc.

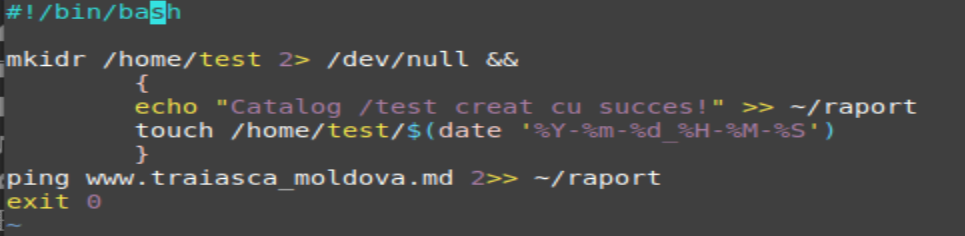
23 0-23/2 \* \* \* cale/către/comandă

Pentru cron există 4 operatori:

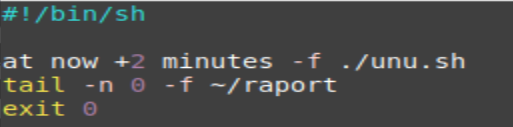
* asteriscul (\*) - specifică toate valorile posibile pentru un câmp; de exemplu, un \* în câmpul oră înseamnă că linia respectivă va fi executată în fiecare oră, un \* în câmpul zi a lunii semnifică execuția în fiecare zi;
* virgula (,) - acest operator specifică o listă de valori; de exemplu, dacă avem în câmpul zi a lunii 1,5,9,17,23,28, linia respectivă se va executa doar în zilele specificate din fiecare lună;
* liniuța (-) - acest operator specifică un interval de valori; de exemplu, putem scrie 1-5 în loc de 1,2,3,4,5;
* separatorul slash (/) - specifică pasul valorilor respective; de exemplu, cum am avut mai sus 0-23/2, semnifică faptul că linia se va executa din 2 în două ore; tot la fiecare două ore mai poate fi specificat prin \*/2; la fiecare 20 de minute putem specifica dacă trecem în câmpul minute \*/20.
* Tratarea semnalelor se realizează prin asocierea unei funcții (handler) unui semnal. Funcția (rutină, handler) reprezintă o secvență de cod, care va fi apelată în momentul în care procesul recepționează semnalul respectiv. Semnalele de sistem au handler-ele sale, de obicei. Pentru prelucrarea semnalului utilizatorului, evident utilizatorul va scrie şi o funcţie ce va prelucra acest semnal.
* Pentru prelucrarea semnalelor în sh (bash), este utilizată comanda trap:
* trap action signal
* Comanda are doi parametri: acțiunea (action) executată la recepționarea semnalului și semnalul pentru care va fi efectuată acțiunea specificată. În mod obișnuit, acțiunea este indicată prin apelarea funcției (cod de script) scrise anterior apelului.
* Cu comanda trap puteți să schimbați handler-ul pentru unele semnale de sistem (cu excepția celor ale căror interceptare este interzisă). În acest caz, handler-ul executat este indicat de argumentul action în comanda trap.

Realizarea sarcinilor.

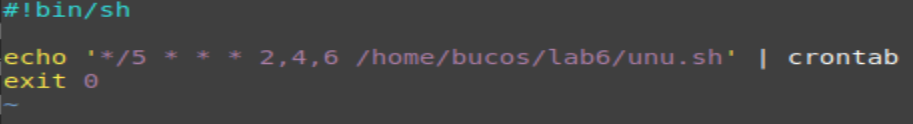
1. Creați și executați o singură dată script-ul (în acest script, nu puteți utiliza operatorul de condiție și operatorii de control a proprietăților și valorilor), care va încerca să creeze directorul test în directorul home. În cazul în care directorul va fi creat, script-ul va scrie în fișierul ~/raport un mesaj "catalog test was created successfully" şi va crea în directorul test un fișier numit Data\_Ora\_Lansarii\_Scriptului. Apoi, indiferent de rezultatele etapei anterioare, script-ul trebuie să interogheze, folosind comanda ping, adresa www.traiasca\_moldova.md și în cazul în care host-ul nu este disponibil, adăugați un mesaj de eroare în fișierul ~/raport (se recomanda - >,>>, ~, date, ||, &&).



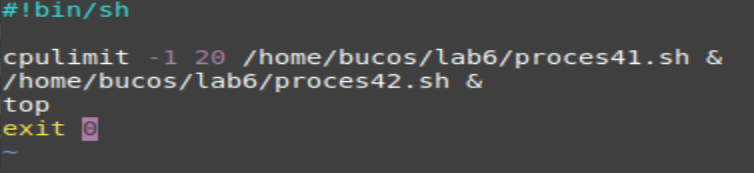
1. Modificați script-ul din 1) pentru încă o singură executare peste 2 minute. Controlați conținutul fișierului ~/raport și afișați liniile noi apărute ( se recomanda – at, tail).



1. Modificați script-ul din 1) ca să ruleze la fiecare 5 minute din oră, în zilele pare ale săptămânii.



1. Creați două procese de fundal care efectuează același ciclu infinit de calcul (de exemplu, înmulțirea a două numere). După lansarea proceselor, prevedeți posibilitatea de a utiliza consolele virtuale din care au fost lansate. Folosind comanda top, analizați procentul de utilizare a microprocesorului de către aceste procese. Rezervați primului proces lansat o rată de utilizare a resurselor microprocesorului nu mai mare de 20% (se recomanda – nice, kill).



Concluzie:

In aceasta lucrare de laborator am studiat comenzile destinate gestionării proceselor, programarea timpului de start al proceselor, transferul și gestionarea datelor între procese