UNIVERSITAT ROVIRA I VIRGILI

Laboratori 3: Processos

Arey Ferrero Ramos

8 de març del 2022

Índex

LastMeu.sh

Solució

Per a poder fer Accounting serà necessari tenir instal·lat el servei acct. Per comprovar si està instal·lat s'utilitza la comanda `systemctl status acct`, que mostra que el servei no està instal·lat. Per instal·lar el servei s'utilitza la comanda `sudo apt-get install acct`. Executant de nou la primera comanda, es comprova que ara el servei està instal·lat i actiu.

S'ha implementat l'script següent.

```
#!/bin/bash


# Autor: Arey Ferrero Ramos.

# Data: 8 de març del 2022. Versió: 1.

# Descripció: Donada una combinació qualsevol de fins a quatre opcions, es mostra diferent informació d'accounting emparant les comandes lastcomm i getopt (implementada en la bash).

#     Paràmetres:

#          -u: Usuari.

#          -c: Comanda.

#          -d: Data.

#          -f: Flag (S, F, D, X).

#     Sortida:

#          -u: Comandes executades per un usuari.

#          -c: Usuari i data en que s'ha executat una comanda.

#          -d: Comandes executades a partir d'una determinada data.

#          -f: Comandes que tenen actiu aquest flag.


if [ $(id -u) -eq 0 ]

then

     if [ $# -lt 9 ]

     then
```

```
lastMe="lastcomm --forwards --strict-match"
activeDate=0
activeFlag=0
while getopts u:c:d:f: option
do
      case $option in
            u)
                  lastMe="$lastMe --user ${OPTARG}"
                  ;;
            c)
                  lastMe="$lastMe --command ${OPTARG}"
                  ;;
            d)
                  if [ $(echo ${OPTARG} | cut -c1-4) -eq
                  $(date +%Y) ]
                  then
                        activeDate=1
                        date=$(echo ${OPTARG} | cut -c5-
                        8)
                  else
                        echo -e "No hi ha informació
                        d'accounting d'anys diferents de
                        l'actual." >&2
                        exit 4
                  fi
                  ;;
            f)
                  activeFlag=1
                  flag=${OPTARG}
                  ;;
            *)
                  echo -e "Error: El paràmetre és
                  incorrecte." >&2
                  exit 3
```

```
                        ;;
                esac
        done
        if [ $activeDate -eq 1 ]
        then
                echo -e "La data no s'ha aconseguit implementar."
                #while read line;
                #do
                        #echo $line | column -t
                        #d=$(echo $linia | awk '{print $(NF-1), $(NF-2)}')
                        #if [ date -d"$d" +%m%d -le $date ]
                        #then
                                #echo $line
                        #fi
                #done < (echo "$lastMe")
        elif [ $activeFlag -eq 1 ]
        then
                $lastMe | grep -E -e "(S|F|C|D|X| )$flag(S|F|C|D|X| )" | awk '{print $1, $(NF-7), $(NF-2), $(NF-1), $NF}' | column -t
        else
                $lastMe | awk '{print $1, $(NF-7), $(NF-2), $(NF-1), $NF}' | column -t
        fi
        exit 0
    else
        echo -e "Error: El número de paràmetres és incorrecte." >&2
        exit 2
    fi
else
    echo -e "Error: Aquest script s'ha d'executar com a root." >&2
    exit 1
```

```
fi
```

Joc de proves

Prova	Descripció	Solució	Correcte?
1	L'usuari no es root.	Error: Aquest script s'ha d'executar com a root.	Sí.
2	El nombre de paràmetres és 5 (-u -c -d -f -u).	Error: El número de paràmetres és incorrecte.	Sí.
3	El únic paràmetre és incorrecte (-a).	Error: El paràmetre és incorrecte.	Sí.
4	Algun dels paràmetres és incorrecte. (-u -c -l).	Error: El paràmetre és incorrecte.	Sí.
5	Execució sense cap paràmetre.	S'executa la comanda lastcomm sense aplicar cap selecció i s'imprimeix la comanda, l'usuari i la data.	Sí.
6	Execució especificant un usuari milax (-u milax).	S'imprimeixen totes les comandes especificades per l'usuari milax i la data en que es van executar.	Sí.
7	Execució especificant una comanda (-c ls).	S'imprimeix els usuaris que han executat la comanda ls i la data en que la van executar.	Sí.
8	Execució especificant un flag (-f F)	S'imprimeixen les comandes que tenen el flag F actiu.	Sí.
9	Execució especificant un usuari i una comanda (-u root -c ls).	S'imprimeixen totes les execucions de la comanda ls per l'usuari root i la data en que van tenir lloc.	Sí.
10	Execució especificant un usuari i un flag (-u milax -f X).	S'imprimeixen totes les comandes executades per l'usuari milax que tenen actiu el flag X i la data en que es van executar.	Sí.
11	Execució especificant una comanda i un flag (-c dpkg -f S).	S'imprimeix totes les execucions de la comanda dpkg que tenen el flag S actiu (cosa que serà sempre) i la data en que van tenir lloc.	Sí.
12	Execució especificant un usuari una comanda i un flag (-u root -c dpkg -f S).	S'imprimeixen totes les execucions de la comanda dpkg fetes per l'usuari root i que tenen el flag S actiu i la data en que van tenir lloc.	Sí.
13	Execució especificant un any anterior a l'actual.	No hi ha informació d'accounting d'anys diferents de l'actual.	Sí.
14	Execució especificant l'any actual.	La data no s'ha aconseguit implementar.	No.

Autoavaluació de la feina realitzada

Com es pot apreciar, no s'ha aconseguit desenvolupar la funcionalitat que imprimeix les comandes executades a partir d'una determinada data. El motiu pel qual no he pogut desenvolupar aquesta funcionalitat ha estat que per a desenvolupar l'exercici s'ha pres una decisió de disseny equivocada. Aquesta decisió ha estat tractar de seleccionar les opcions corresponents als paràmetres utilitzant les pròpies opcions de la comanda lastcomm enlloc de fent servir la comanda grep. Per desenvolupar la data es requeria utilitzar un bucle i tal com s'ha implementat no hi havia manera d'aconseguir que aquest bucle tractés el contingut de la variable lastMe (Variable que emmagatzema la comanda lastcomm i els seus paràmetres) com una comanda a executar enlloc de com un string. En el moment en el que vaig decidir renunciar a tractar d'implementar aquest disseny sota la conclusió de que és impossible, ja era massa tard per redissenyar l'exercici.

Degut a que soc plenament conscient de que d'haver utilitzat la comanda grep des de bon principi es probable que hagués aconseguit resoldre l'exercici i que això a més m'hagués deixat més temps per acabar de polir algun exercici posterior, la veritat es que no em sento gaire disposat a posar-me nota d'aquest exercici ja que no el puc valorar de forma objectiva com he fet amb els altres.

Protecció del procés lastMeu.sh dels senyals SIGINT i SIGTERM

Solució

Per a que un procés generat al executar el procés lastMeu.sh pugui bloquejar un senyal s'ha d'utilitzar la comanda `trap`. Concretament, per evitar que el procés pugui ser avortat amb el senyal SIGINT s'utilitza la comanda `trap 'echo "No es pot avortar el procés amb el senyal SIGINT."' SIGINT` i per evitar que el procés pugui ser avortat amb el senyal SIGTERM s'utilitza la comanda `trap 'echo "No es pot avortar el procés amb el senyal SIGTERM."' SIGTERM`.

Com es pot veure, aquesta comanda permet especificar una instrucció que serà executada quan el senyal especificat s'enviï al procés. S'ha decidit utilitzar aquesta funcionalitat per a construir el joc de proves. Així, s'ha definit una variable anomenada `variable`, amb una sentencia inicial (`variable="No s'ha intentat avortar el procés."`). Aquesta variable passarà a contenir un missatge informatiu relacionat amb el senyal SIGINT quan aquest s'enviï (`trap 'variable="No es pot avortar el procés amb el senyal SIGINT."' SIGINT`) i un missatge informatiu relacionat amb el senyal SIGTERM quan aquest s'enviï (`trap 'variable="No es pot avortar el procés amb el senyal SIGTERM."' SIGTERM`). Degut a que l'script s'executa durant un període de temps massa curt com per a poder fer proves d'execució, s'ha

afegit al final la següent estructura iterativa que imprimeix `variable` un nombre infinit de vegades.

```
while true
do
      echo $variable
done
```

D'aquesta manera, si la comanda trap s'ha executat de manera correcta, es podrà comprovar com canvia el valor de `variable` segons el senyal enviat.

Quan s'ha acabat de fer les proves d'execució s'han eliminat aquesta estructura iterativa i s'ha canviat la instrucció especificada a la comanda trap per la comanda d'impressió per pantalla mostrada inicialment.

Joc de proves

Prova	Descripció	Solució	Correcte?
1	Es crea el procés executant l'script lastMeu.sh.	S'imprimeix 'No s'ha intentat avortar el procés'.	Sí.
2	Es prem la combinació de tecles CTRL + C.	S'imprimeix 'No es pot avortar el procés amb el senyal SIGINT.'.	Sí.
2	S'envia el senyal SIGINT al procés amb PID 16957 amb la comanda `kill -SIGINT 16957`.	S'imprimeix 'No es pot avortar el procés amb el senyal SIGINT.'.	Sí.
3	S'envia el senyal SIGTERM al procés amb PID 16957 amb la comanda `kill -SIGTERM 16957`.	S'imprimeix 'No es pot avortar el procés amb el senyal SIGTERM.'.	Sí.
4	Es torna a enviar el senyal SIGINT al procés amb PID 16957 amb la comanda `kill -SIGINT 16957`.	S'imprimeix 'No es pot avortar el procés amb el senyal SIGINT.'.	Sí.
5	Es torna a enviar el senyal SIGTERM al procés amb PID 16957 amb la comanda `kill -SIGTERM 16957`.	S'imprimeix 'No es pot avortar el procés amb el senyal SIGTERM.'.	Sí.

Autoavaluació de la feina realitzada

S'ha realitzat tot el que es demanava en aquest exercici. A més, l'explicació ha estat clara i el disseny del joc de proves ha estat molt net. Per aquest motiu la nota que hauria de treure és un 10.

aturarSenseFi.sh, continuarSenseFi.sh i programació periòdica

Solució

aturarSenseFi.sh

```
#!/bin/bash


# Autor: Arey Ferrero Ramos.
# Data: 8 de març del 2022. Versió: 1
# Descripció: S'aturen una vegada els processos anomenats
senseFi (Processos que s'han creat cada cert temps i en hores
diferents i que consumeixen CPU i no acaben mai).
#     Paràmetres:
#          -
#     Retorn:
#          -Fitxer amb els PIDs dels processos que es vagin
aturant.


if [ $# -eq 0 ]
then
      IFS=$'\n'
      for proces in $(ps aux | grep senseFi | head -n -1)
      do
            if [ $(echo $proces | tr -s ' ' | cut -f8 -d' ') =
            "R+" ]
            then
                  pid=$(echo $proces | tr -s ' ' | cut -f2 -d' ')
                  kill -SIGSTOP $pid
```

```
                echo $pid >> pidsProcessos.txt
            fi
        done
elif [ $1 = "-h" ]
then
        echo -e "aturarSenseFi.sh: S'aturen una vegada els
        processos anomenats senseFi (Processos que s'han creat cada
        cert temps i en hores diferents i que consumeixen CPU i no
        acaben mai).\n\tParàmetres:\n\t\t-\n\tRetorn:\n\t\t-Fitxer
        amb els PIDs dels processos que es vagin aturant." >&2
        exit 1
else
        echo -e "Error: Els paràmetres son incorrectes." >&2
        exit 2
fi
```

Per a que aquest script s'executi a cada hora en punt, s'ha afegit al fitxer del servei crontab amb la comanda `sudo echo "0 * * * * root /home/milax/GSX/GestionSistemas/Lab3/aturarSenseFi.sh" >> /etc/crontab`.

continuarSenseFi.sh

```
#!/bin/bash

# Autor: Arey Ferrero Ramos.
# Data: 8 de març del 2022. Versió: 1
# Descripció: A les 21:30 es reanuda l'execució d'un conjunt de
processos recollits en un fitxer, amb una diferencia de temps de
30 segons entre cada un.
#     Paràmetres:
#          -Fitxer amb els PIDs dels processos que s'han anat
aturant.
#     Retorn:
#          -
```

```
if [ $# -eq 1 ]
then
	if [ -f $1 ]
	then
		IFS=$'\n'
		for pid in $(cat $1)
		do
			kill -SIGCONT $pid
			sleep 30
		done
	elif [ $1 = "-h" ]
	then
		echo -e "continuarSenseFi.sh: A les 21:30 es reanuda
		l'execució d'un conjunt de processos recollits en un
		fitxer, amb una diferencia de temps de 30 segons
		entre cada un.\n\tParàmetres:\n\t\t-Fitxer amb els
		PIDs dels processos que s'han anat
		aturant.\n\tRetorn:\n\t\t-" >&2
		exit 1
	else
		echo -e "Error: El paràmetre '$1' és incorrecte." >&2
		exit 2
	fi
else
	echo -e "Error: El nombre de paràmetres és incorrecte." >&2
	exit 3
fi
```

Per a que aquest script s'executi a partir de dos quarts de les 9 de la nit, s'ha afegit al fitxer del servei crontab amb la comanda `sudo echo "30 21 * * * root /home/milax/GSX/GestionSistemas/Lab3/continuarSenseFi.sh" >> /etc/crontab`.

Joc de proves

aturarSenseFi.sh

Prova	Descripció	Solució	Correcte?
1	S'introdueix un paràmetre incorrecte (-a).	Error: Els paràmetres son incorrectes.	Sí.
2	Opció d'ajuda (-h).	aturarSenseFi.sh: S'aturen una vegada els processos anomenats senseFi (Processos que s'han creat cada cert temps i en hores diferents i que consumeixen CPU i no acaben mai). Paràmetres: - Retorn: -Fitxer amb els PIDs dels processos que es vagin aturant.	Sí.
3	Es creen tres processos senseFi i s'executa l'script (sense paràmetres). Es llista el fitxer amb la comanda `cat pidsProcessos.txt`.	S'aturen els processos. El fitxer pidsProcessos s'ha creat correctament i conté els PIDs dels tres processos que s'han creat (15897, 15942, 15948).	Sí.
4	Es creen tres processos senseFi més i s'executa l'script (sense paràmetres). Es llista el fitxer amb la comanda `cat pidsProcessos.txt`.	S'aturen els processos. El fitxer pidsProcessos s'ha creat correctament i conté els PIDs dels tres processos que s'han creat (15897, 15942, 15948, 16415, 16422, 16427).	Sí.
5	S'afegeix l'script aturarProcessos.sh al fitxer `/etc/crontab` per a comprovar si s'executa a una hora programada ().	No s'aturen els processos.	No.

continuarSenseFi.sh

Prova	Descripció	Solució	Correcte?
1	Cap paràmetre.	Error: El nombre de paràmetres és incorrecte.	Sí.
2	Dos paràmetres (-a -b).	Error: El nombre de paràmetres és incorrecte.	Sí.

3	Paràmetre incorrecte (-a).	Error: El paràmetre '-a' és incorrecte.	Sí.
4	Opció d'ajuda (-h).	continuarSenseFi.sh: A les 21:30 es reanuda l'execució d'un conjunt de processos recollits en un fitxer, amb una diferencia de temps de 30 segons entre cada un. Paràmetres: -Fitxer amb els PIDs dels processos que s'han anat aturant. Retorn: -	Sí.
5	S'executa l'script i se li passa com a paràmetre el fitxer `pidsProcessos.txt`.	Es reanuda l'execució dels processos que estaven aturats.	Sí.
6	S'afegeix l'script continuarSenseFi.sh al fitxer `/etc/crontab` per a comprovar si s'executa a una hora programada ().	No es reanuda l'execució dels processos.	No.

Autoavaluació de la feina realitzada

No s'ha aconseguit implementar tot el que es demanava en l'exercici. Els dos scripts funcionen perfectament quan s'executen de forma immediata però no quan s'executen de forma periòdica amb el daemon del cron. Tot i així, la comanda que s'ha proposat per afegir els scripts al fitxer /etc/crontab i que aquests s'executin de forma periòdica a l'hora indicada se sap que és correcta perquè es va utilitzar en el laboratori anterior. D'aquí el que es pot deduir és que aquest problema està relacionat amb la variable PATH. Una nota raonable tenint tot això en compte podria ser un 8.

Execució dels processos senseFi en un mateix cgrup amb una sola CPU

Solució

El motiu pel qual es vol que tots els processos senseFi de l'exercici anterior siguin executats dins d'un mateix cgroup és per a poder restringir la seva execució a una única CPU. Per tant, el directori a on s'haurà d' accedir és el `/sys/fs/cgroup/cpuset` per

al qual es requereixen permisos de root. Dins d'aquest directori es crea el cgroup grupSenseFi amb la comanda `mkdir /sys/fs/cgroup/cpuset/grupSenseFi`. Degut al funcionament d'aquesta funcionalitat, en aquest directori ja hi estaran creats tots els fitxers que es puguin necessitar, tot i que no contenen informació.

Per a aconseguir el nostre objectiu, caldrà destinar una CPU i un bloc de memòria als processos que s'agrupin dins del cgrup grupSenseFi. Per a saber quantes CPUs hi ha disponibles en el nostres sistema s'haurà d'examinar el fitxer `/sys/fs/cgroup/cpuset/cpuset.cpus` i per saber quants blocs de memòria hi ha disponibles en el nostres sistema el fitxer `/sys/fs/cgroup/cpuset/cpuset.mems`. Executant la comanda `cat /sys/fs/cgroup/cpuset/cpuset.cpus` es comprova que en el sistema hi ha 4 CPUs numerades del 0 al 3 i amb la comanda `cat /sys/fs/cgroup/cpuset/cpuset.mems` es comprova que en el sistema hi ha un únic bloc de memòria 0. Es destina la CPU 0 als processos senseFi amb la comanda `echo 0 >> /sys/fs/cgroup/cpuset/grupSenseFi/cpuset.cpus` i el bloc de memòria 0 (l'únic bloc de memòria disponible) amb la comanda `echo 0 >> /sys/fs/cgroup/cpuset/grupSenseFi/cpuset.mems`.

Finalment, obrim tres terminals per crear en cada una un procés senseFi, però amb l'objectiu de que s'executin en el mateix cgrup. Per a que això passi, els PIDs dels tres processos han d'estar en el fitxer `/sys/fs/cgroup/cpuset/grupSenseFi/tasks`. La manera més intuïtiva d'aconseguir això seria crear cada procés en la seva respectiva terminal i després afegir el PID al fitxer. El problema de fer-ho d'aquesta manera es que el processos no s'estaran creant dins del mateix cgrup des de l'inici. Així, la manera de fer-ho és afegir al fitxer `/sys/fs/cgroup/cpuset/grupSenseFi/tasks` el procés pare del procés senseFi, que és la terminal. Per obtenir el PID de cada terminal, s'executa la comanda echo $$ en cada terminal i s'afegeix el PID obtingut al fitxer `/sys/fs/cgroup/cpuset/grupSenseFi/tasks` amb les comandes `echo 11079 >> /sys/fs/cgroup/cpuset/grupSenseFi/tasks`, `echo 22381 >> /sys/fs/cgroup/cpuset/grupSenseFi/tasks` i `echo 22381 >> /sys/fs/cgroup/cpuset/grupSenseFi/tasks`.

Ara, qualsevol procés que es creï en qualsevol d'aquestes tres terminals formarà part del cgrup senseFi, cosa que es pot comprovar llistant el contingut del fitxer `/sys/fs/cgroup/cpuset/grupSenseFi/tasks` després de l'execució del procés.

Joc de proves

Prova	Descripció	Resultat	Correcte?
1	Es destina la CPU 0 als processos senseFi. Es comprova si es correcte executant la comanda `cat`	Es mostra la CPU destinada. 0	Sí.

	`sys/fs/cgroup/cpuset/grupSenseFi/cpuset.cpus`.		
2	Es destina el bloc de memòria 0 als processos senseFi. Es comprova si es correcte executant la comanda `cat sys/fs/cgroup/cpuset/grupSenseFi/cpuset.mems`.	Es mostra el bloc de memòria destinat. 0	Sí.
3	S'afegeixen els PIDs de les tres terminals al cgrup grupSenseFi. Es comprova si es correcte executant la comanda `cat sys/fs/cgroup/cpuset/grupSenseFi/tasks`.	Es mostra els PIDs de les tres terminals. 11079 11136 11149	Sí.
4	S'executa un procés senseFi en cada una de les tres terminals. Es comprova si es correcte executant la comanda `cat sys/fs/cgroup/cpuset/grupSenseFi/tasks`.	Es mostra els PIDs de les tres terminals y dels tres processos senseFi. 11079 11136 11149 32616 32624 32629	Sí.

Autoavaluació de la feina realitzada

S'ha realitzat tot el que es demanava en aquest exercici. Per aquest la nota hauria de ser un 10.