PRACTICAS DE SISTEMA OPERATIVO UNIX

PRÁCTICAS DE UNIX

PRACTICA 7. ***Procesos***

Objetivos
En esta práctica se realizarán ejercicios de control de procesos en las modalidades foreground y background, así como tratamiento de prioridades, evaluación de tiempos y planificación de trabajos.

Herramientas
Las herramientas a utilizar serán:
ps jobs bg fg kill nice sleep time at nohup wait stty stop

DESARROLLO DE LA PRACTICA

1. Procesos foreground.

1.1 Averigüe cual es la actividad actual del sistema. Para ello visualice un listado completo del estado de todos los procesos que se están ejecutando en el sistema.

```
ps -e
```

1.2 Obtener un listado con los siguientes datos de los procesos de su shell actual.

```
PID     PPID      PRIORIDAD   NICE    COMANDO
...     ...       ...         ...     ...
...     ...       ...         ...     ...
```

El listado debe salir ordenado por pid.

```
ps -l | tr -s" " | cut -f4,5,7,8,14 -d" "  | sort
```

1.3. Cree un alias llamado "prio" que informe del PID, PPID, prioridad, valor nice, TTY y comando asociados a todos los procesos del sistema cuyo propietario sea usted.

```
alias prio=pl -l | tr -s" " | cut -f4,5,7,8,12,14 -d" "
```

1.4 Indique la/s diferencia/s en el resultado de las siguientes instrucciones. Explique el por qué.

```
$ (who && who | wc -l) | tee quien
$ who && who | wc -l | tee quien
```

La primera manda al archivo 'quien' y a pantalla la lista de usuarios conectados y el número de ellos, la segunda manda a pantalla todo, pero solo el número de usuarios conectados al fichero
La causa, es que el entubamiento hacia tee del segundo caso, solo afecta al segundo operando del &&.

1.5 Indique las diferencias en la ejecución de las siguientes instrucciones:

a. $ (cd .. ; pwd)
b. $ { cd .. ; pwd } (después de las llaves hay un espacio en blanco.)
c.
```
$ {
cd ..
pwd
}
```

b y c se ejecutan en un subshell.

1.6 La orden "kill $$":

a. ¿Cuál es su objetivo?
Eliminar el shell actual.
b. ¿Lo consigue?
No.

c. Modifique la instrucción para conseguir dicho fin.
kill -9 $$

1.7 Con una única instrucción, obtenga el proceso o procesos que tenga(n) mayor valor de nice.
```
ps -l | sort -t" " +8 | head -1
```

2. Prioridad subordinada. Procesos background

2.1 Cree un shell script "shell_71" cuyo contenido sean las dos instrucciones siguientes:
```
sleep 100
find / -name "passwd" -exec grep `logname` {} \; >> resultado
2> /dev/null
```
Ejecútelo como proceso subordinado (background).
```
shell_71&
```

2.2 Ejecute el guión otras dos veces más modificando la prioridad en cada ejecución, primero disminuyéndola en 5 unidades y después en 10. Verifique los diferentes valores nice asignados, para lo cual ejecute la orden "ps -l".
```
nice -5 shell_71&
nice -10 shell_71&
ps -l
```

2.3 Interrumpa el proceso correspondiente al segundo shell lanzado. Utilice inmediatamente después, el comando "ps" para ver el grado de éxito obtenido.
```
kill -9 <PID>
```

2.4 ¿Qué diferencias encuentra al ejecutar las siguientes cuatro instrucciones?

a. `(sleep 50 ; sleep 55)&`
b. `sleep 60 ; sleep 65 &`
c. `sleep 70& sleep 75&`
d. `sleep 80&;sleep 85&`

a, c y d mandan los sleep en background. El primer sleep de b se ejecuta en foreground.

2.5 Ejecute el fichero "shell_71" tres veces cada una en background pero utilizando una sola línea.
```
shell_71&;shell_71&;shell_71&;
```

2.6 Ejecute el comando:
```
find  / -name ".profile" - user `logname` -exec cat {} \;
```
calculando el tiempo que tarda en ejecutarse.
Ahora haga lo mismo en background dejando el resultado en un fichero llamado "miprofile", evitando posibles mensajes de error.

2.7 Cree un guión "shell_72" que almacene en una variable "ord" el número de archivos ordinarios que tenga en la estructura de directorios y en otra variable "dir" el número de directorios. Las instrucciones para conseguir lo anterior se deben lanzar en background.
A continuación se visualizarán los mensajes:
```
Número de ficheros ordinarios <ord>
Número de directorios <dir>
Total ... <xxx>
```
Asegúrese de que no se visualizarán los mensajes hasta que no se hayan terminado de ejecutar las instrucciones subordinadas.

```
ord={ find / -type f -depth }& > $i1
dir={ find / -type d -depth }& > $i2
until [ ! `ps $i1` && ! `ps $i2` ]
do
sleep 1
done
echo "Ficheros ordinarios: $ord"
echo "Directorios        : $dir"
echo "TOTAL              : `expr $ord + $dir`"
```

2.8 Ejecute el shell anterior pero evitando paradas imprevistas. Pruebe a abandonar el sistema y retornar, comprobando que el proceso activado prosigue su ejecución. Indique en qué fichero la orden "nohup" almacena los resultados y los posibles errores.
`nohup.out`

2.9 Cierre la sesión de trabajo, inicie una nueva y verifique que solo tiene asociados los procesos correspondientes a su nueva sesión. Ejecute las siguientes instrucciones:
$ sleep 600 &
$ sleep 700 &
$ ps -f > pslist1
$ kill -9 $$
Abra una nueva sesión de trabajo. ¿Qué ha sucedido con los dos procesos hijos del proceso de conexión anterior, generados con las órdenes sleep?
Compruébelo de la siguiente manera:
1. Ejecute la orden "`ps -f -u `logname` > pslist2`"
2. Compare los archivos "pslist1" y "pslist2" y saque conclusiones.

(No tarde más de 10 minutos en realizar esta prueba ya que 600/60=10)

2.10 Ejecute la instrucción "$find / -name "fichero" 2> /dev/null &"
Recuerde el número de proceso que se ha asociado a la instrucción anterior.
Convierta el proceso background anterior en un proceso foreground.
`stop <PID>`
`fg <PID>`

2.11 Verifique si tiene activada la suspensión de trabajos para lo cual utilice la orden "`$ stty`" y busque en el listado que se presenta la cadena "`susp`" para ver que secuencia de caracteres tiene asociada (por ejemplo puede aparecer "`susp = ^Z`" lo cual indica que pulsando ctrl+z se suspende la ejecución de un trabajo foreground).
Si no tiene asociada una combinación de teclas asigne una, por ejemplo con la orden "`$ stty susp ^Z`".
a. Ejecute la orden "$ sleep 400" en foreground.
b. Suspenda su ejecución con la combinación de teclas en cuestión.
c. Mire cuál es el número de proceso del trabajo suspendido y convierta dicho proceso en un proceso subordinado (background).

`bg <PID>`

3. Planificación de trabajos

3.1 Indique una instrucción para que visualice dentro de diez minutos el mensaje:
"Han pasado diez minutos."
`{sleep 600;echo "Han pasado 10 minutos";}&`

3.2 Planifique un trabajo para que se ejecute el día 5 de diciembre a las 9 de la mañana. Este trabajo será la ejecución de un guión que se encontrará en su directorio de conexión y cuya misión será informar de que los días 6 y 8 de diciembre son festivos. Cree el guión e indique la instrucción para su planificación.
```
at 0900 12,05
aviso
wall "Los dias 6 y 8 de diciembre son festivos."
```

3.3 Planifique otro trabajo que le recuerde la fecha de su cumpleaños lo cual conseguirá enviando en dicha fecha a su correo un mensaje de "felicidades".
```
at 0000 11,07
mail f941162 < echo "Felicidades!!"
```

3.4 Visualice los trabajos planificados con la herramienta "at".
`at -l`

3.5 Con una única instrucción, elimine el trabajo planificado anteriormente para el día 5 de diciembre, sin efectuar una consulta previa de la referencia del trabajo en cuestión.
`at -r`