

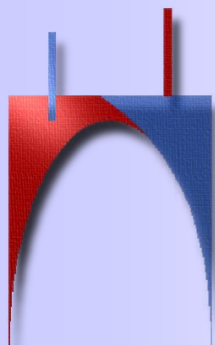
C.F.G.S.

Desarrollo de Aplicaciones Multiplataforma

Desarrollo de Aplicaciones Web

UD 5.2

Sistemas Operativos Multiusuario. Procesos



Instituto de Educación Secundaria
Santiago Hernández
Informática

Introducción

➤ Definiciones

- Proceso: Petición de un programa al sistema para ser ejecutado
- Daemon: Proceso del sistema (se ejecuta sin solicitud del usuario)
- Hilos (Threads): Partes de un proceso que pueden ejecutarse en paralelo

➤ Conceptos

- PID: Identificador de Proceso. Asignado por el Sistema
- Siempre existe al menos un proceso en ejecución
- Shell: un proceso
- Un proceso puede crear otros (padre y hijos).
- Un padre: varios hijos. Un hijo: solo un padre.
- Morir: Dejar de ejecutarse. Deben morir todos los hijos.

➤ Variables

- \$\$: PID del shell actual
- \$!: PID del ultimo proceso lanzado en segundo plano



Control de Procesos

top

muestra los programas en ejecución.

➤ **top [opciones]**

- Interactivo
- h ó ?: ayuda
- q: salir

ps

muestra los programas en ejecución.

➤ **ps [-f|-l] [opciones]**

Por defecto muestra los programas ejecutados en este terminal por el usuario que lo invoca

- listado normal: PID, TTY, TIME, CMD
- **-f** (listado largo 1): UID(nombre), PID, PPID, C, STIME, TTY, TIME, CMD(completo)
- **-l** (listado largo 2): F, S, UID(número), PID, PPID, C, PRI, NI, ADDR, SZ, WCHAN, TTY, TIME, CMD
- Otras opciones:
 - **-e** (todos los procesos)
 - **-u** lista (procesos de esos usuarios (UID o nombre))
 - **-p** lista (esos procesos (PID))
 - **-t** lista (esos terminales (ver man))

- F: Indicador en octal del estado del proceso (man ps)
- S: Estado del proceso.
 - O: en ejecución
 - S: Dormido
 - R: Preparado
 - Z: Zombie
- UID: Usuario que lanza el proceso
- PID: Identificador de proceso
- PPID: Identificador del proceso padre
- C: Estimación de uso de la CPU
- PRI: Prioridad (mayor número=menor prioridad)
- NI: Número NICE
- ADDR: Dirección del proceso en la tabla de procesos
- SZ: Tamaño que puede intercambiarse (swap)
- WCHAN: Si dormido: dirección del proceso en la tabla de procesos
- STIME: hora o fecha de comienzo de ejecución
- TTY: terminal asociado (? indica que no tiene terminal asociado)
- TIME: Tiempo acumulado en el procesador
- CMD: nombre del comando

kill

Envía señales a los procesos por su PID

➤ **kill [-señal] número_proceso [...]**

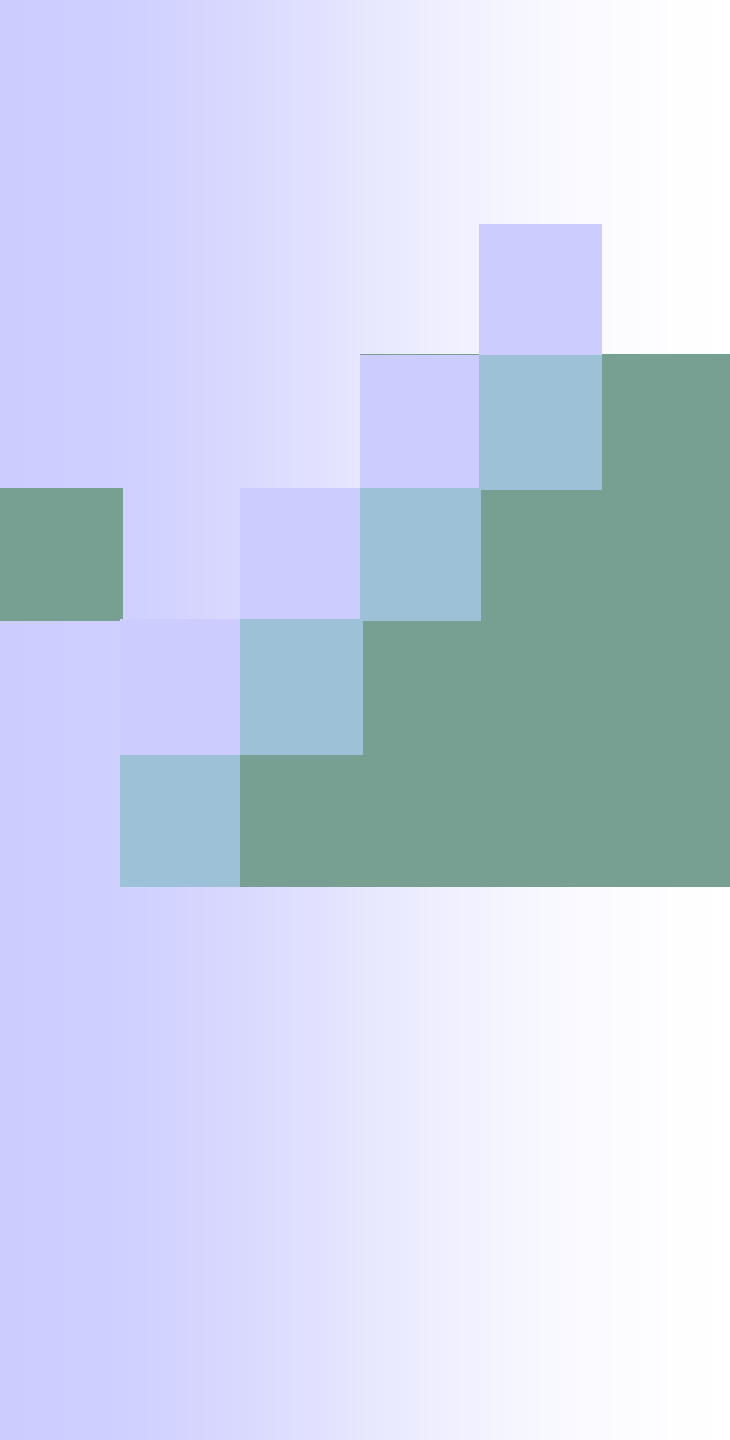
- Si no se indica señal, se manda la señal 15 (TERM: parar)
- -9: (KILL: matar) (no puede ser ignorada ni atrapada por el proceso)
- -l: lista de señales

killall

Envía señales a los procesos por su nombre

➤ **killall [-señal] nombre_proceso [...]**

- Enviará la señal a todos los procesos que tengan ese nombre
- Por defecto se manda la señal 15
- -l: lista las señales



Ejecución de Procesos

Ejecución en primer plano (foreground)

➤ Invocación

- Se ejecuta un pipeline, toma control del terminal y no puede ejecutarse otro hasta que ha terminado
- El pipeline se ejecuta, habitualmente, en el shell actual (el entorno queda modificado)
- Puede forzarse la creación de un shell hijo escribiendo el pipeline entre paréntesis (el entorno no queda modificado)

(pipeline)

➤ Invocación secuencial

- Se separan los pipelines por ";"
- Al terminar de ejecutarse uno, comienza el siguiente.

pipeline1 ; pipeline2
(pipeline1 ; pipeline2)

➤ Invocación condicional

- Se ejecuta el pipeline de la izquierda
- El siguiente se ejecutará según haya quedado el valor de la variable de estado

`pipeline1 && pipeline2`

- se ejecuta pipeline1. Si el la variable de estado queda en 0 (se suele entender que ha hecho algo de forma correcta o que ha encontrado lo que queríamos) se ejecutará pipeline2

`pipeline1 || pipeline2`

- se ejecuta pipeline1. Si el la variable de estado no queda en 0 (se suele entender que ha cometido un error o que no ha encontrado lo que queríamos) se ejecutará pipeline2

Ejecución en segundo plano (background)

➤ Invocación

- Se ejecuta un pipeline, sus acciones se ejecutarán pero antes de terminar (normalmente de forma inmediata) se devuelve el control al shell para que pueda seguir trabajándose en él.
- Si durante la ejecución en background se produce alguna salida a stdout o stderr esta lo hará normalmente por lo que es recomendable (por no decir imprescindible) redireccionarlas
- Para lanzar un pipeline (completo) a segundo plano el último carácter debe ser "&"
pipeline &
- El shell nos responderá indicándonos cual es el número de proceso en segundo plano y devolviendo el control
- Cuando el proceso termine el shell lo indicará de forma no intrusiva (cuando pueda hacerlo sin molestar o después de un return si estamos en el prompt del shell)

➤ **jobs**

- Comando solo disponible en algunos shell
- Muestra la lista de procesos en segundo plano lanzados desde el terminal actual e indica cual es el más moderno y el anterior)
- Además indica el estado del proceso:
 - Done
 - Running
 - Stopped

➤ **Cambio de plano**

- De primer a segundo plano: pulsando **^Z**
- De segundo a primer plano: ejecutando **fg n°** (el número es el que aparece al ejecutar jobs)

➤ **Reanudar trabajos en segundo plano**

- **bg n°** (el número es el que aparece al ejecutar jobs)

➤ nohup

- Impide que se mate un proceso en segundo plano al hacer un logout
 - Si lanzamos procesos a segundo plano sin el nohup, al hacer un logout deberemos escribir exit dos veces seguidas (a la primera no hace nada)
 - Si el logout se produce con éxito, al morir el proceso del shell, mueren todos sus hijos y por tanto se morirán todos los procesos que estén en segundo plano.
- Sintaxis:
 nohup pipeline [&]
- stdout se redirecciona automáticamente a ./nohup.out (o \$HOME/nohup.out si no se tiene permiso de escritura en aquel)



Control de tiempo y sincronización

sleep

produce una pausa para sincronizar procesos

➤ **sleep *n*[m|h|d]**

- produce una pausa de ***n*** segundos (o minutos, horas o días si se añade la letra correspondiente)

time

muestra el tiempo que tarda en ejecutarse un pipeline

➤ time pipeline

➤ /usr/bin/time [-p] pipeline

puede estar en otro directorio, comprobarlo con whereis

- La opción **-p** hace que la información salga en el mismo formato que en la forma 1
- Según las distribuciones o, incluso, el shell la salida puede producirse directamente a pantalla (por lo que no es redireccionable) o a stdout. Puede ser incluso diferente entre una forma y otra dentro de la misma distribución y shell



Prioridad

Introducción

- Prioridad: importancia que tiene un proceso.
- Se entiende que los procesos con mayor prioridad se ejecutarán antes que otros con menor prioridad, que se les dará más tiempo de procesador (u otros recursos) o ambas cosas.
- Todos los procesos se lanzan con una prioridad establecida por el sistema operativo según sus parámetros de configuración.
- Los procesos pueden ser lanzados con otra prioridad o incluso ser modificada una vez que tienen creado su PCB.
- En unix la prioridad se obtiene sumando dos valores: La prioridad en sí y el número nice. Éste número es modificable por los usuarios.
- A mayor número resultante, menor prioridad tendrá el proceso.

nice

Lanza un proceso con una prioridad diferente

➤ **nice [-valor] pipeline**

- Lanza el pipeline con menor prioridad

➤ **nice --valor pipeline**

- Lanza el pipeline con mayor prioridad (solo root)
- El valor de reducción de la prioridad si no se indica es variable entre distribuciones o sistemas
- El rango en que puede modificarse el número nice también depende de las distribuciones

renice

Modifica la prioridad de un proceso en ejecución

➤ **renice [-]valor -p PID | -u UID [...]**

- -valor: aumenta prioridad (solo root)
- +valor: disminuye prioridad
- los rangos, al igual que nice, son variables según la distribución
- de la misma manera en comportamiento de renice es diferente, en unos casos aumenta o disminuye el valor del numero nice en el valor indicado y en otros (como es el caso de suse) coloca el número nice en el valor indicado (siempre y cuando sea mayor al existente o seamos root)



Planificación

Introducción

- Planificar: Ejecutar procesos sin estar necesariamente sentados físicamente en la máquina
- La planificación está controlada por demonios (crond y atd). Tendremos que asegurarnos que están en ejecución para que funcione correctamente.
- Las salidas producidas por los procesos ejecutados periódicamente (cron) se pierden salvo que se redirecciones a un fichero
- Las salidas producidas por los procesos diferidos (at) se redireccionan hacia el correo (mail) si no se hace manualmente hacia un fichero
 - Tienen que estar instalados:
 - un servicio de correo (p.e. postfix)
 - En fedora, como root o con sudo:
 - yum install postfix
 - systemctl enable postfix
 - systemctl start postfix
 - un programa que permita gestionarlos (p.e., mailx)
 - En fedora, como root o con sudo:
 - yum install mailx

Ejecución periódica

➤ crontab [-u usuario] opción

- Opciones:
 - **-u usuario** (solo root, para editar los trabajos planificados por cualquier usuario)
 - **-l** (listar el fichero)
 - **-e** (editar el fichero)
 - **-r** (borrar el fichero)
- El fichero se modifica con vi
- Si se borra el fichero ese usuario no tendrá trabajos periódicos
- Cada línea tiene seis columnas:
 1. Minuto (0..59) *=60 veces por hora
 2. Hora (0..23) *=24 veces por día
 3. Día del mes (1..31) *=ignora y utiliza campo 5
 4. Mes (1..12) *=12 veces al año
 5. Día de la semana (0..6) *=ignora y utiliza campo 3 (0=domingo)
 6. Comando a ejecutar

* en campos 3 y 5 = 7 veces a la semana
- En los campos 1 a 5 pueden utilizarse rangos (inicio-final) y/o listas separadas por comas

- **Consejos:**

- Suele ser un error poner * en el campo 1
- No se deben programar varios trabajos a la misma hora
- Es recomendable no utilizar las horas exactas
- Ojo con las noches. Algo que queramos ejecutar en la madrugada del viernes al sábado tendrá el valor 6 en día de la semana

- **Ejemplos:**

- 30 1 * * 2-6 comando
- 0 12 * 1,3,5,7-12 0 comando

Ejecución diferida

➤ at hora [fecha] [incremento]

- Hora:
 - 1, 2 ó 4 dígitos (1 o dos dígitos: solo hora. 4 dígitos: hh:mm)
 - hh:mm
 - horas especiales: now, noon, midnight
- Fecha:
 - mes (3 primeras letras del nombre en inglés) seguido de día y, opcionalmente, ",año"
 - día de la semana (nombre completo ó 3 primeras letras del nombre en inglés)
 - días especiales: today, tomorrow
- Incremento:
 - +n unidades (minutes, hours, days, weeks, months, years o 3 primeras letras)
- Ejemplos:
 - at now +1 min
 - at 1200 jul 25,2035
- Al ejecutar el comando, el shell espera que introduzcamos una a una las líneas de los comandos a ejecutar. Cuando terminemos pulsaremos ^D y quedará programado
 - ubuntu 16.04 no instalado por defecto: apt-get install at
 - Opensuse 42, Fedora 26 (systemd): puede ser necesario instalar y activar el servicio atd

➤ **batch**

- Igual que at pero sin indicar cuando queremos ejecutarlo.
- El Sistema ejecutará los comandos cuando estime conveniente.

➤ **Control de la ejecución diferida**

- Pueden controlarse los trabajos programados con at y batch
 - at -l (lista los trabajos)
 - at -r número (borra el trabajo indicado)

Control de la planificación

➤ crontab

- Ficheros:
 - cron.allow (lista de usuarios que pueden utilizarlo)
 - cron.deny (lista de usuarios que no pueden utilizarlo)
 - Ubicación: variable (utilizar el find)

➤ at y batch

- Ficheros:
 - at.allow (lista de usuarios que pueden utilizarlo)
 - at.deny (lista de usuarios que no pueden utilizarlo)
 - Ubicación: variable (utilizar el find)

➤ Política de uso

- Si existe .allow se usa su lista (independientemente de que exista o no .deny)
- Si no existe .allow se usa la lista de .deny (si existe)
- Si no existe ni uno ni otro solo podrá root.