3.1. Processos

Yolanda Alemany Ruiz 2020-2021

Definició

Programa: seqüència d'instruccions, definides a priori, que poden ser executades per un processador.

Procés: programa en execució

Els ordinadors realitzen diverses tasques alhora. En un cert instant, cada CPU només pot executar una instrucció, així que fa canvis ràpids, imperceptibles, entre les diverses tasques que executa -> **multiprogramació**

Això crea una falsa il·lusió de paral·lelisme -> **pseudoparal·lelisme**

Requisits del sistema operatiu

Per a gestionar correctament els processos, el sistema operatiu:

- Ha de **maximitzar la utilització del processador**: màxim de processos, mínim de temps.
- Ha de tenir una **política d'assignació de recursos** als processos.
- Ha de **permetre la comunicació entre processos**.

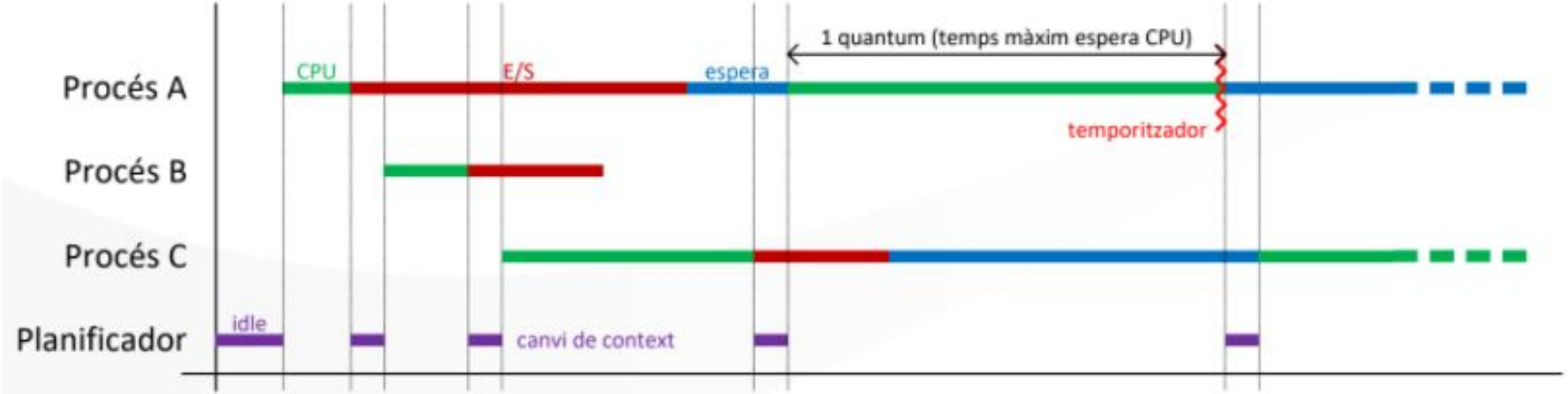
Planificador de processos

La missió del processador és executar les instruccions de la màquina que resideixen a la memòria principal.

El processador **intercala l'execució de processos per raons d'eficiència**. El planificador de processos ho gestiona:

- Aprofita que **el temps d'E/S són molt més alts que els d'execució de la CPU**
- Si un procés porta massa temps executant-se salta el **temporitzador** per a que no monopolitzi el processador.
- Les accions necessàries per a passar d'un procés a un altre s'anomenen **canvi de context**.
- Si en un moment no hi ha res executant-se, s'executa el **procés inactiu (idle)**

Planificador de processos (scheduler)



Prioritats de processos

- No totes les tasques tenen les mateixes exigències pel que fa al temps i disponibilitat de recursos: aquesta gestió es fa mitjançant les **prioritats dels processos**.
- Les prioritats poden ser assignades pel propi **sistema operatiu** o pel **propietari** de la tasca.
- La prioritat d'un procés no es fixa al llarg del temps, i una de les operacions més comunes del sistema operatiu és el canvi de prioritat d'un procés.

Processos a Unix: *Process identifier* (PID)

A Unix i Windows tot procés s'identifica amb un nombre únic, **el PID**.

El sistema operatiu disposa d'una taula de processos on cada procés és identificat pel seu PID.

- **Una aplicació no pot crear un procés directament!** és necessari demanar al sistema operatiu que crei el procés i el gestioni. Es disposa de dos crides de sistema:
  - **fork**: permet que un procés crei un altre procés. Aquest nou procés es anomenat procés fill.
  - **exec:** permet executar una aplicació externa. Substitueix el procés en execució per un altre procés.

Tipus de processos

- **Procés fill**: creats pel **fork**.
- **Procés orfe:** el pare ha finalitzat la seva execució abans que el fill.
- **Procés zombie**: procés que ha acabat però està a l'espera d'una resposta del pare.
- **Procés aturat**: es poden aturar els processos
- **Procés dimoni (servei)**: s'executa permanentment i en segon terme (background). No té interfícies d'usuari associades.

Tipus de processos

- **Nucli (kernel):** el primer procés en executar-se.
  - És executat pel gestor d'arrancada
  - El procés del nucli que gestiona tota la resta de processos s'anomena **scheduler (planificador de processos).** No es pot visualitzar (té el **PID 0**).
- **Procés init**
  - És el procés **pare de tota la resta de processos d'usuari** (té el **PID 1**).
  - Si un procés queda sense pare (orfe), el procés init l'adoptarà.
- **Procés kthreadd**
  - És el procés **pare dels processos que únicament executen codi del Kernel**. Són especials (no segueixen les regles de consum de memòria, etc). (té el **PID 2**).
  -

Comanda pstree - Linux

Executeu: pstree -p | head

- També hi ha l'opció $ ps -elf -- forest

La comanda pstree permet visualitzar la jerarquia de processos:

- Localitzeu l'interpret d'ordres on heu executat pstree.
- Executeu $ sleep 25 &
- Torneu a executar pstree
- Vegeu com el procés sleep és fill de bash i germà de pstree

Creació de processos - Linux

- **Primer pla (foreground - fg)**: és el mode per defecte amb el qual executem ordres a l'interpret d'ordres. Les ordres bloquejen l'execució de l'interpret.

  $ sleep 10

- **Segon pla (background - bg):** es pot fer amb el símbol & (o l'ordre bg)

  $ sleep 10 &

  - Ens proporciona el PID
  - Es per poder gestionar el procés
  - Per exemple, matar el procés amb un kill.

Creació de processos - Windows

- **Primer pla (foreground - fg)**: és el mode per defecte amb el qual executem ordres a l'interpret d'ordres. Les ordres bloquejen l'execució de l'interpret. (Powershell)

  Start-Process calc
  Start-Process -FilePath 'c:\Windows\System32\calc.exe'

- **Segon pla (background - bg)**: utilitzem l'ordre **Start-Job**

  Start-Job {calc}

Control i monitorització de tasques - Linux

- **ps**: mostra la taula de processos del terminal i usuari actual.
- **jobs**: mostra els processos que s'estan executant a l'intèrpret de comandes (dóna un nombre de tasca a cadascuna)
- **top**: mostra els processos ordenats per percentatge d'ús de CPU (de més a menys) i de forma continuada.
- **fg**: passa un procés de segon terme a primer terme (amb un nombre de tasca/job)
- **bg**: passa un procés de primer terme a segon terme (amb un nombre de tasca/job)

Control i monitorització de tasques - Windows

- **Get-Process**: mostra la taula de processos del terminal i usuari actual
- **Get-Job**: mostra els processos que s'estan executant en segon pla
- **Get-Process | Sort-Object cpu -Descending**: mostra els processos ordenats per percentatge d'ús de CPU (de més a menys).

Matar processos - Linux

- **3 formes d'acabar prematurament un procés que s'està executant en primer pla:**
  - **Ctrl-C**: s'envia el senyal 2 (**SIGINT**). La majoria d'aplicacions estan programades per finalitzar l'execució quan reben aquesta senyal. Algunes comandes no ho fan (requereixen una doble confirmació abans de finalitzar el procés).
  - **Ctrl+\:** s'envia el senyal 3 (**SIGQUIT**). També depèn de com estigui programada l'aplicació.
  - Utilitzar **kill.**
- **L'única forma de finalitzar un procés que s'executa en segon pla és utilitzar la comanda kill.**

Matar processos - Linux

- Només podem matar processos nostres o qualsevol procés si som root.
  - **Kill -9 250** (mata un procés pel seu PID)
  - **Kill -9 -1** (mata tots els processos)
  - **killall mozilla** (mata un procés pel seu nom)

La comanda kill (Linux)

Quan executem la comanda kill, realment no matem un procés, si no que li enviem un senyal determinat.

- El sistema disposa de 4 senyals que s'indentifiquen per un nombre o un nom simbòlic (SIG+Nom_del_senyal).
- Cada procés pot adaptar tots els senyals excepte 2:

  SIGKILL (9): permet finalitzar un procés de forma abrupta.

  SIGSTOP (19): permet aturar l'execució d'un procés.

La comanda kill (Linux)

- A més, hi ha altres senyals habitualment utilitzats:

  INT (2): Interrupt. És enviada quan fem CTRL+C. Els programes poden dicidir què fan en rebre aquest senyal.

  TERM (15): Terminate. Finalitza el procés de forma controlada (per exemple, demanaria guardar els canvis). És el que executa kill per defecte i el que s'envia als programes quan finalitzem el sistema.

  … kill -l ens mostra la taula completa de signals amb els seus noms.

La comanda kill (Windows) &

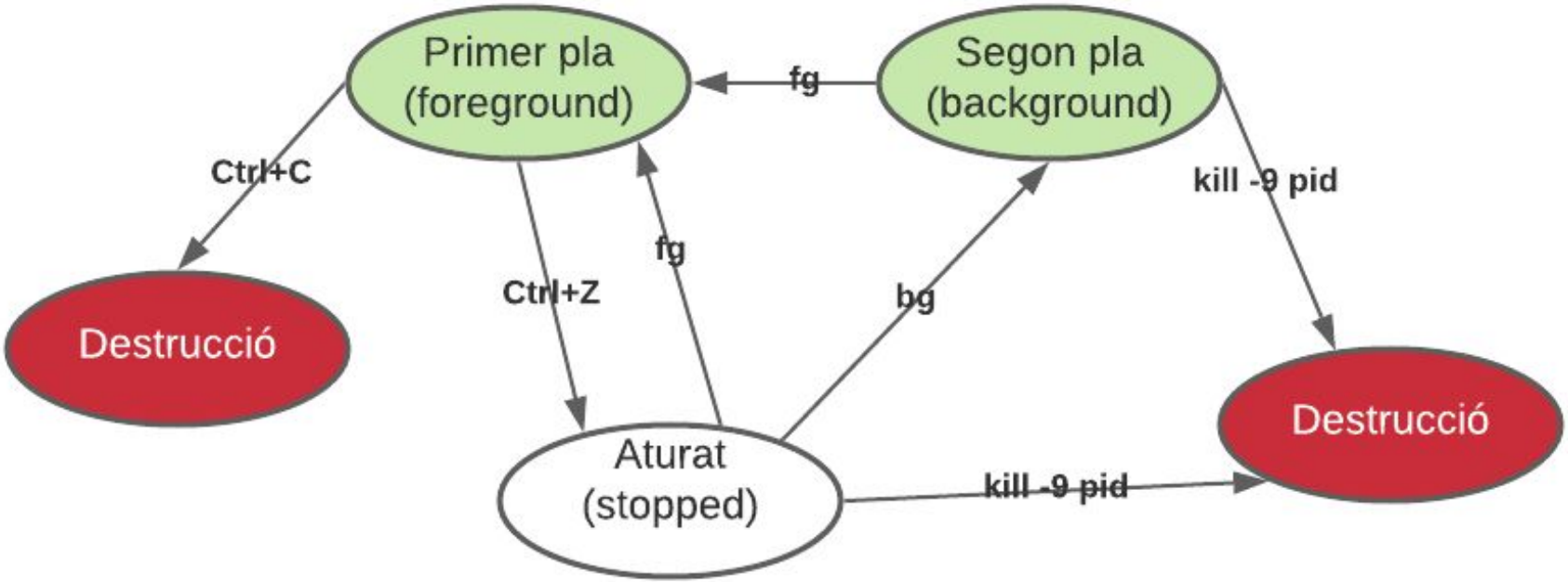
Utilitzem la comanda Stop-Process per matar un procés:

- **Stop-Process-Id 5893** (mata un procés pel seu PID)
- **Stop-Process-Name “Notepad”** (mata un procés a partir del seu nom)
- **Stop-Process-Name outlook, winword, MicrosoftEdge** (mata un procés a partir del seu nom)

Utilitzem la comanda Start-Process per arrancar un procés, és necessari indicar com a paràmetre l’executable del programa:

- **Start-Process calc**
- **Start-Process ‘c:\Program’ Files\MicrosoftOffice\Office15\WINWORD.EXE’**
- **Start-Process iexplore.exe -ArgumentList http://www.cifpfbmoll.eu/**

Estats d'un procés Linux



Prioritats dels processos - Linux

Per indicar la prioritat d'un procés: **Valors del rang: -20 a 19**

- Valor més alt -20 (màxima prioritat)
- Valor més baix 19 (mínima prioritat)
- La prioritat per defecte és 0.

Els processos més prioritaris utilitzen més recursos. Els processos amb prioritat mínima (19), només s'executaran quan el sistema no executa cap altra tasca.

Els usuaris només poden modificar els processos dels quals són propietaris en un internal de 0 a 19.

L'usuari root pot canviar la prioritat de qualsevol procés a qualsevol valor.

Prioritat dels processos Linux - Comandes

- **nice**: permet executar un procés amb una prioritat concreta

  P.e. **nice -n 15 gedit**

- **renice:** permet modificar la prioritat d'un procés en execució

  P.e. **renice +15 785** (modifica la prioritat del procés amb PID=785)

  P.e. **renice +19-u estudiant** (com a superusuari, modifica la prioritat - dels processos d'un usuari determinat)

- **Consultar la prioritat a Windows**

  Get-Process | Format-Table-View priority: consultem la prioritat dels processos (Alta-Normal-Baixa)

Prioritat dels processos Linux - Comandes

El sistema de fitxers muntat a **/proc** mostra tota la informació respecte als processos que hi ha en execució.

- Si us situeu al directori i executeu la comanda **ls** veureu un llistat de fitxers i directoris que ofereixen informació general sobre la màquina.
  - **cpuinfo, meminfo, version…**
- Cada procés està representat per un directori amb el PID del procés.
- Dins d'aquests directoris hi ha informació general sobre el procés:
  - La línia de comandes que l'ha creat: **cmdline**
  - Les variables d'entorn: **environ**
  - L'estat: **status**

Si executeu **man proc** podeu veure informació sobre aquest sistema de fitxers.

Comunicació entre processos

Els diversos processos que s'executen a la màquina es poden comunicar, els uns amb els altres, mitjançant diversos mecanismes.

A continuació veurem com funcionen aquests mecanismes de comunicació i algunes de les eines que més habitualment s'utilitzen en aquest àmbit.

Concatenació d'ordres

Tenim diferents operadors per executar múltiples ordres en una mateixa línia:

Operador AND: només s'executa la comanda 2 si la primera comanda s'ha executat amb èxit

$ mkdir directori && cd directori

Operador OR: només s'executa la comanda 2 si la primera comanda no s'ha executat amb èxit.

$ mkdir directori1 || mkdir directori2

Concatenació d'ordres

Flux de dades (stream):

- És un conjunt d'informació que es desplaça d'un origen a una destinació
- Els flux de dades són bàsics per tal de realitzar tasques complexes combinant programes senzills.
- Internament, el sistema operatiu els tracta com si fossin fitxers.

3 fluxos de dades estàndard a Linux

- **Entrada estàndard (stdin)**: És el flux d'entrada de les ordres i aplicacions (habitualment el teclat).
- **Sortida estàndard (stdout):** És el flux de sortida de les ordres i aplicacions. Normalment és la terminal.

Concatenació d’ordres

- **Error Estàndard (stderr):** Linux proporciona un segon tipus de flux de dades de sortida. La idea és tenir una sortida per aquelles dades amb alta prioritat com, per exemple, els missatges d’error. Per defecte és la terminal.

Filosofia de Linux

Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

Doug McIlroy, creador del conductes Unix

Programes petits amb una sola utilitat.

Les operacions complexes es realitzen combinant programes.

Es treballa amb fluxos de dades de text.

Redireccions

Les redireccions permeten modificar els flux de dades estàndard.

Es pot modificar l'entrada o la sortida estàndard que utilitza una ordre o aplicació.

Per exemple, es pot redireccionar la sortida d'una ordre cap a un fitxer:

```
$ echo Hola!> hola.txt
$ cat hola.txt
Hola!
```

2 tipus principals: Redirecció de la sortida **>** i redirecció de l'entrada: **<**

Operadors de redirecció del **bash / PowerShell**

Operador	Descripció
>	Crea un nou fitxer contenint la sortida estàndard (no la d'error!) de la comanda executada. Atenció, si el fitxer existeix es sobreescrit (i es perden les dades existents).
>>	Afegeix (append) a un fitxer la sortida la sortida estàndard. Si el fitxer no existeix, el crea. Les noves dades s'afegeixen al final del fitxer i no s'eliminen les dades existents.
2>	Crea un nou fitxer contenint la sortida d'error estàndard de la comanda executada. Atenció, si el fitxer existeix es sobreescrit (i es perden les dades existents).
2>>	Afegeix (append) a un fitxer la sortida d'error estàndard. Si el fitxer no existeix es crea. Les noves dades s'afegeixen al final del fitxer i no s'eliminen les dades existents.

Operadors de redirecció del **bash / PowerShell**

Operador	Descripció
&>	Crea un nou fitxer contenint la sortida estàndard i la sortida d'error estàndard de la comanda executada. Atenció, si el fitxer existeix es sobreescrit (i es perden les dades existents).
<	Envia les dades del fitxer especificat a la dreta com a entrada estàndard de la comanda de l'esquerra.
<<	Accepta un text de diverses línies com a entrada estàndard.
<>	El fitxer especificat a la dreta és tant l'entrada estàndard com la sortida estàndard de la comanda de l'esquerra.

Operadors de redirecció del **bash / PowerShell**

Existeix un fitxer molt especial que serveix per a silenciar la sortida d'una comanda: **/dev/null (forat negre).**

A Windows utilitzem la comanda 'Out-Null' (Ex: **Get-Command | Out-Null**)

https://linuxhint.com/what_is_dev_null/

Conductes (pipes)

Els conductes són un mecanisme que permet enllaçar la sortida estàndard d'una ordre amb l'entrada estàndard d'una altra ordre.

Permeten concatenar l'execució de diverses comandes:

$ comanda1 | comanda2 | comanda3 | comanda4

- La sortida de comanda1 serà l'entrada de comanda2.
- La sortida de comanda2 serà l'entrada de comanda3.
- La sortida de comanda3 serà l'entrada de comanda4.
- La sortida de comanda4 es mostrarà al terminal.

Comanda **tee** (**T**)

Dividir la sortida estàndard per tal que sigui mostrada per pantalla i guardada en tants fitxers com s'indiquin.

```
$ ls -la | tee sortidaLs.txt
```

Mostrarà el resultat de **ls -la** i el guardarà també al fitxer **sortidaLS.txt**

Per defecte tee copia la seva entrada estàndard sobre la seva sortida estàndard i, a més, sobreescriu els fitxers que indiquem com a paràmetres (elimina els continguts previs si el fitxer ja existia).

Comandes **echo** i **sleep**

En aquest bloc farem servir la comanda **echo** en diverses ocasions, la qual permet escriure per la sortida estàndard.

```
$ echo “sistemes Operatius”
PS Write-Host ‘Sistemes Operatius’
PS echo ‘Sistemes Operatius’
```

Una altra comanda que trobarem freqüentment serà **sleep**, que permet adormir o aturar l’execució durant el nombre de segons que es passin com a paràmetre.

```
$ sleep 10
PS Start-Sleep 10
```

Comandes **cat**, **more**, **less**, **pg** i **tac**

cat és l'abreviatura de **concatenate** s'utilitza per concatenar fitxers.

```
$ cat colors animals
vermell
blau
verd
pingüí
moix
gavina
```

S'acostuma a combinar amb paginadors:

- more: cat/etc/passwd/etc/shadow | more
- less: cat/etc/passwd/etc/shadow | less
- pg: cat/etc/passwd/etc/shadow | pg

- La comanda **tac** permet fer el mateix, ordenant inversament.
- Utilitzem **cat** i **more** a PowerShell

Comandes **head** i **tail**

Sovint només volem consultar les primeres o últimes línies d'un fitxer:

head: mostra les primeres 10 línies d'un fitxer. Amb el paràmetre **-n** podem indicar quantes línies volem mostrar.

- **head /etc/passwd** és equivalent a **car /etc&passwd | head**
- **ls | head -n 20**

PS Get-Content prova.txt | Select-Object -First 5

PS Get-Content prova.txt --Head 5

tail: mostra les últimes 10 línies d'un fitxer. Amb el paràmetre **-n** podem indicar quantes línies volem mostrar.

- ls | tail -n 5

PS Get-Content prova.txt | Select-Object -Last 5

PS Get-Content prova.txt -Tail 5

Comanda **sort** (Linux)

La comanda **sort** disposa de moltes opcions que permeten ordenar les línies d'un fitxer d'entrada, segons el criteri que indiquem.

- Podem ordenar alfabèticament o numèricament, ascendentment o descendentment, definir la part de la línia per ordernar (camp), determinar quin delimitador utilitzem per a separar els camps (per defecte, l'espai).

```
$ sort /etc/password
$ sort -t: -k3-n /etc/passwd
$ ls -l /etc | sort -k5 -n -r | more
```

Comanda **sort** (Linux)

wc: per defecte, retorna el nombre de línies, paraules i bytes d'un fitxer. Disposem de paràmetres per a comptar només els caràcters, bytes, línies o paraules.disposa de moltes opcions que permeten ordenar les línies d'un fitxer d'entrada, segons el criteri que indiquem.

```
$ wc /etc/passwd
$ wc -c /etc/passwd
$ ls -l / | wc -l
```

nl: numera les línies d'un fitxer, amb diverses opcions pel que fa a aquesta numeració.

```
$ cat quijote.txt | nl
$ nl quijote.txt
```

Comanda **sort** (Linux)

pr: prepara un fitxer per a la seva impressió: afegeix pàgines, capçaleres i peus.

```
$ cat quijote.txt | pr
```

Comanda **Measure-Object** (Windows)

Measure-Object (equivalent a wc): per defecte, retorna el nombre de línies, paraules, caràcters, etc.

```
PS Get-Content prova.txt | Measure-Object -Line
PS Get-Content prova.txt | Measure-Object -Word
```

Equivalent a nl: numera les línies d'un fitxer

```
PS Select-String .* -Path quijote.txt | Select Linenumber, Line
```

Comandes **fmt** i **uniq**

fmt: formata línies llargues, segons la mida (en caràcters) que se li indiqui. Per defecte limita a 75 caràcters per línia, però es pot modificar. Té en compte els espais.

```
$ fmt quijote.txt
$ ls -l | fmt -w 10
```

uniq: compacta en una única línia totes les línies **repetides** i **consecutives.** Disposa d'altres opcions de compactació. Es sol combinar amb **sort** per tal d'aconseguir el resultat anterior.

```
$ uniq numeros.txt
```

Comanda **Get-Unique** (Windows)

El paràmetre **AsString**, em permet tractar les dades rebudes per la PIPE (|) com un text i no com un objecte.

```
PS C:\> 1,1,1,1,12,23,4,5,4643,5,3,3,3,3,3,3,3 | Sort-Object | Get-Unique
PS Get-Process | Sort-Object | Select-Object processname | Get-Unique -AsString
```

Comanda **split**

Permet dividir un fitxer en altres fitxers. Podem triar si volem fraccionar el fitxer:

- En funció d'una mida determinada de bytes
- En funció d'un nombre de caràcters
- En funció d'un nombre de línies

```
$ ls -l > prova.txt
$ split -l 1 prova.txt linia
$ cat linia*
```

Comanda **split**

La comanda **split** no accepta que li passem el contingut a dividir per l'entrada estàndard.

```
$ ls --l | split -l 1 linia
```

Per això utilitzem el guió (-), que permet utilitzar l'entrada estàndard com si fos un fitxer:

```
$ ls -l | split -l 1 - linia
```

Comandes **paste** i **join**

paste: permet fusionar les línies de diversos fitxers, en un de sol, línia a línia. Utilitza el tabulador com a separador.

```
$ paste lletres.txt numeros.txt
```

join: permet fusionar les línies de dos fitxers, respecte un camp:

```
$ join usuaris.txt contrasenyes.txt
```

Comanda **cut**

Selecciona una porció de totes les línies d'un fitxers d'entrada.

- Podem especificar la porció de la línia com a rang de caràcters (no recomenable) o com a rang de camps (com per a camps específics, guió per a rangs).

```
$ date | cut -c11-18
$ cut -d: -f3,5 --output-delimeter="" /etc/passwd
```

- Es poden indicar rangs oberts (fins al final / des del principi).
- Podem indicar altres paràmetres com el delimitador de sortida.
- També se li pot indicar que faci el complement del que es retalla (tot menys el que es demana a la comanda).