Chapitre 1 :

Gestion et Ordonnancement des processus

1

1

Plan

- La notion de processus
- Gestion des processus

2

2

La notion de processus

3

Processus: Introduction

- Programme et processus

Programme

Un programme est une suite d'instruction que le système doit exécuter au processeur pour résoudre un problème. La suite d'instructions est rangée dans un fichier.

Processus

Un processus est un programme en train d'être exécuté par le système dans un environnement donné

4

Processus: Introduction

Processus

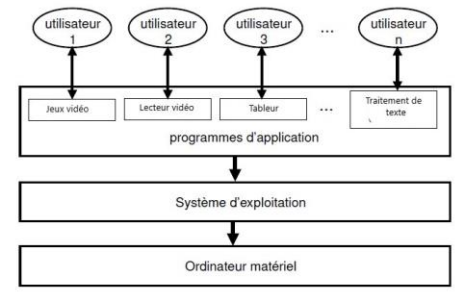
- Tous les ordinateurs modernes sont capables de faire plusieurs choses simultanément
- Par exemple: Exécuter un programme utilisateur, écrire sur le disque dur, lancer une impression (système multiprogrammé)
- Le processeur bascule entre les différents programmes à exécuter (chacun s'exécutant quelques dizaines de millisecondes) ⟹ pseudo-parallélisme
- Modèle conceptuel reposant sur les processus séquentiels

5

5

Processus: Introduction

- Différence entre processus et programme
  - Le programme est une description statique
  - Le processus est une activité dynamique avec
    - un début, un déroulement et une fin
    - Il possède un état qui évolue au cours du temps
- Le processus représente l'abstraction d'un programme en cours d'exécution. Il possède un programme, des données en entrée et en sortie ainsi qu'un état courant



6

6

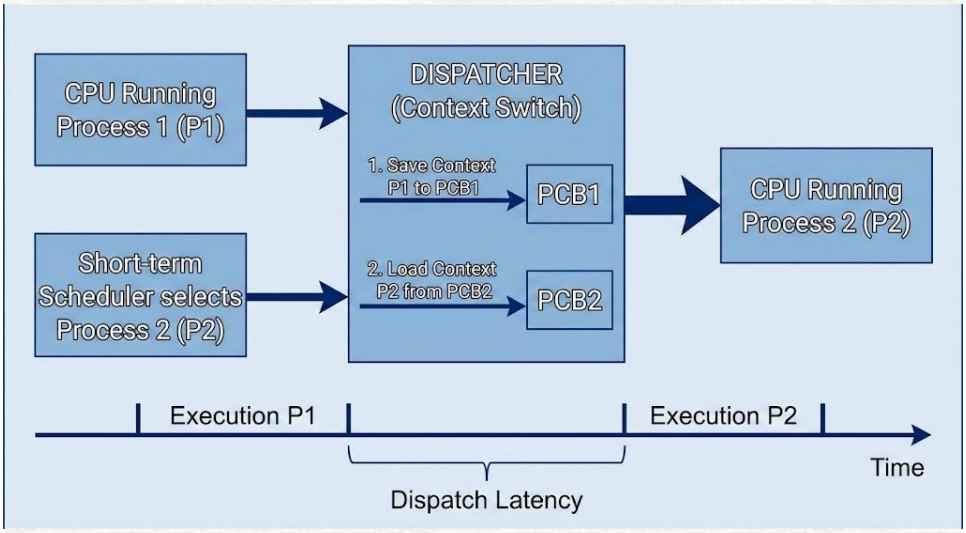
Processus: Introduction

- Le processeur est considéré comme la ressource la plus importante dans une machine, donc il convient de bien gérer ce dernier afin de le rendre plus productif.
- Pour atteindre cet objectif, les concepteurs des systèmes d'exploitation délèguent l'allocation du processeur à un module du système d'exploitation en l'occurrence le *Dispatcheur*.
- Ce Dispatcheur est exécuté chaque fois que le processus en cours d'exécution se bloque ou se termine, outre ces deux situations il peut être également exécuté en cas de réquisition du processeur si l'ordonnancement prend en considération cette propriété.

7

7

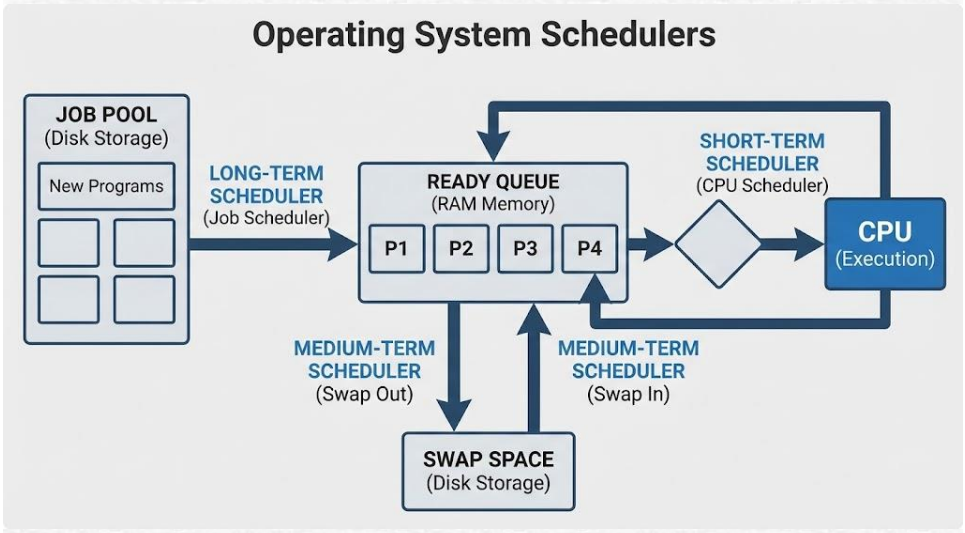
Processus: Introduction



8

8

Processus: Introduction



9

9

Gestion des processus

10

10

Processus: création d'un processus

- Sous Unix/Linux les processus forment une hiérarchie:
  - Chaque processus à un processus père
  - Un processus peut avoir 0, 1, n processus fils
- Seul le processus init, le processus qui initialise le système n'a pas de processus père
- Les événements susceptibles de créer un processus
  - Initialisation du système
  - Création par appel système d'un processus au cours de l'exécution d'un autre processus
  - Requête utilisateur de création d'un nouveau processus (par l'intermédiaire d'un interpréteur de commandes)

11

11

Processus: Modes de Fonctionnement

- On distingue deux modes de fonctionnement

En avant-plan (foreground)

- Le père attend la terminaison du processus fils avant de lancer un autre processus
- Les processus s'exécutent en séquence
- L'utilisateur peut interagir avec le processus en cours d'exécution (saisie de données)
- Mode par défaut

En arrière-plan (background)

- Le père n'attend pas que le fils ait terminé pour continuer à travailler ;
- Ces processus sont le plus souvent des serveurs en attente de requêtes (par exemple serveur d'impression, serveur de messagerie, …) ;
- Les processus s'exécutent en parallèle

12

12

Processus: modes de fonctionnement

- Exemples de lancement de processus: gérer les taches dans le shell
  - Lorsque des processus sont lancés depuis un terminal, certains shells modernes et en particulier le bash, fournissent un ensemble de mécanismes pour gérer leur exécution. Dans ce contexte, on parle de tâches (jobs)

```
saad@saad:~$ gedit
^C
saad@saad:~$ gedit &
[1] 20200
saad@saad:~$
```

13

13

Processus: Interruption et fin d'un processus

Interruption d'un processus

➥ Un processus peut être interrompu pendant son exécution

- en attente d'un périphérique
- en attente de la CPU
- en attente d'un événement logiciel

14

14

Processus: Interruption et fin d'un processus

Interruption d'un processus

➥ Un processus peut être interrompu pendant son exécution

- en attente d'un périphérique
- en attente de la CPU
- en attente d'un événement logiciel

Fin d'un processus

① Arrêt normal (volontaire à l'aide de l'appel système `exit(0)`)
② Arrêt contrôlé par l'algorithme pour erreur
③ Arrêt non contrôlé par l'algorithme pour erreur
④ Arrêt provoqué par l'intervention d'un autre processus

15

15

Processus: Gestion des processus

Table des processus

- Le système maintient une table pour gérer l'ensemble des processus
- Chaque processus est représenté dans cette table par un numéro d'identification : le **pid**
- Chaque entrée de la table correspond aux informations sur le processus

16

16

Processus: Gestion des processus

Table des processus

- Le système maintient une table pour gérer l'ensemble des processus
- Chaque processus est représenté dans cette table par un numéro d'identification : le **pid**
- Chaque entrée de la table correspond aux informations sur le processus

Une entrée de la table

- le numéro d'identification du processus père : **ppid**
- l'identifiant de l'utilisateur qui exécute le processus : **uid**
- l'identifiant du groupe de l'utilisateur qui exécute le processus : **gid**
- le répertoire courant
- la liste des fichiers utilisés par le processus
- le masque de création des fichiers
- le terminal de contrôle associé
- la zone mémoire associée
- ...

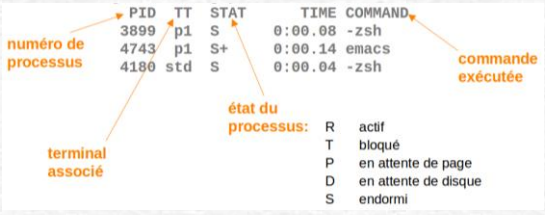
17

17

Processus: Gestion des processus

- La commande **ps** permet de visualiser tous les processus en cours
- Pour chaque processus listé, affiche:
  - le pid
  - le terminal associé
  - le temps CPU
  - le nom de la commande tapée

```
saad@saad:~$ ps
    PID TTY          TIME CMD
   3093 pts/0    00:00:00 bash
  20200 pts/0    00:00:00 gedit
  20213 pts/0    00:00:00 ps
saad@saad:~$ ps
    PID TTY          TIME CMD
   3093 pts/0    00:00:00 bash
  20223 pts/0    00:00:00 ps
[1]+  Done                    gedit
saad@saad:~$
```

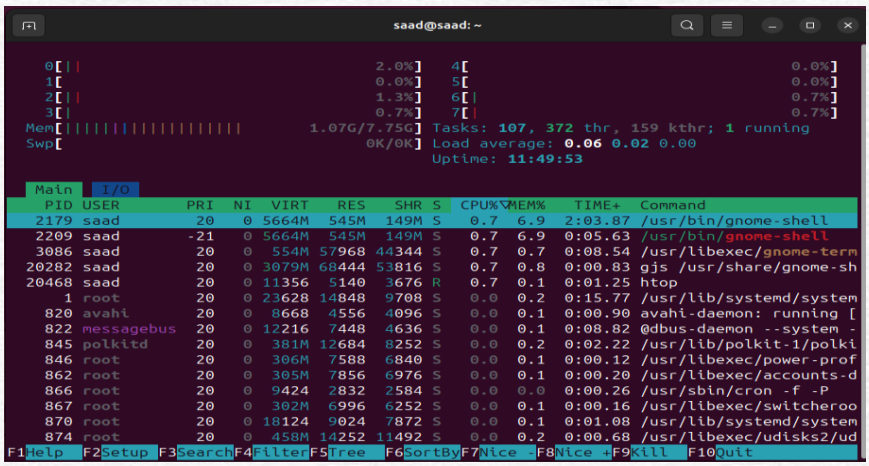


18

18

Processus: Gestion des processus

- La commande **htop** offre une vue interactive avec un code couleur pour faciliter la gestion des processus.



19

19

Processus: Gestion des processus

- Des options permettent d'obtenir plus d'informations

Exemples options

- a présente les processus des autres utilisateurs
- u présente le nom de l'utilisateur et l'heure de lancement
- x affiche les processus qui ne sont pas associés à un terminal de contrôle

```
saad@saad:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1  23628 14848 ?        Ss   Jan22   0:15 /usr/lib/systemd/system
root           2  0.0  0.0      0     0 ?        S    Jan22   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    Jan22   0:00 [pool_workqueue_release
root           4  0.0  0.0      0     0 ?        I<   Jan22   0:00 [kworker/R-rcu_gp]
root           5  0.0  0.0      0     0 ?        I<   Jan22   0:00 [kworker/R-sync_wq]
root           6  0.0  0.0      0     0 ?        I<   Jan22   0:00 [kworker/R-kvfree_rcu_r
root           7  0.0  0.0      0     0 ?        I<   Jan22   0:00 [kworker/R-slub_flushwq
root           8  0.0  0.0      0     0 ?        I<   Jan22   0:00 [kworker/R-netns]
root          13  0.0  0.0      0     0 ?        I<   Jan22   0:00 [kworker/R-mm_percpu_wq
root          14  0.0  0.0      0     0 ?        I    Jan22   0:00 [rcu_tasks_kthread]
root          15  0.0  0.0      0     0 ?        I    Jan22   0:00 [rcu_tasks_rude_kthread
root          16  0.0  0.0      0     0 ?        I    Jan22   0:00 [rcu_tasks_trace_kthrea
root          17  0.0  0.0      0     0 ?        S    Jan22   0:00 [ksoftirqd/0]
root          18  0.0  0.0      0     0 ?        I    Jan22   0:15 [rcu_preempt]
root          19  0.0  0.0      0     0 ?        S    Jan22   0:00 [rcu_exp_par_gp_kthread
root          20  0.0  0.0      0     0 ?        S    Jan22   0:02 [rcu_exp_gp_kthread_wor
```

20

20

Processus: Gestion des processus

- La commande **pstree:** C'est un outil puissant qui affiche les processus en cours d'exécution sous forme d'arborescence. Cette représentation visuelle facilite la compréhension de la hiérarchie des processus, offrant une vue plus intuitive des relations entre les processus.

```
saad@saad:~$ pstree
systemd─┬─ModemManager───3*[{ModemManager}]
        ├─NetworkManager───3*[{NetworkManager}]
        ├─accounts-daemon───3*[{accounts-daemon}]
        ├─avahi-daemon───avahi-daemon
        ├─colord───3*[{colord}]
        ├─cron
        ├─cups-browsed───3*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─fwupd───5*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b───3*[{gnome-session-b}]
        │      │                 │                 └─3*[{gdm-wayland-ses}]
        │      │                 └─3*[{gdm-session-wor}]
        │      └─3*[{gdm3}]
        ├─gnome-remote-de───3*[{gnome-remote-de}]
        ├─2*[kerneloops]
        ├─packagekitd───3*[{packagekitd}]
```

21

21

Processus: Gestion des processus

- Rappel des commandes
- **Cat : affiche le contenu d'un fichier**
  - Syntaxe : cat [-n] fichier
  - Option principale : -n : numérote les lignes
  - NB : Exemples :
  - cat -n essai.1 : affiche le fichier essai.1 à l'écran en numérotant les lignes.
  - cat essai.1 essai.2 : affiche à l'écran le fichier essai.1 puis le fichier essai.2.
- **More : affiche le contenu d'un fichier page par page.**
  - Syntaxe : more [-c] fichier
  - Option principale : -c : efface la fenêtre avant l'affichage.
  - Exemple :
  - more essai.c : affiche la première page du fichier essai.c. Puis tapez : espace pour afficher la page suivante. [Enter] pour afficher la ligne suivante. h pour accéder à l'aide en ligne. q pour quitter.

22

22

Processus: Gestion des processus

- **Less: affiche le contenu d'un fichier page par page.**
  - Syntaxe : less [-c] fichier
  - Option principale : -c : efface la fenêtre avant l'affichage.
- **Head: affiche le début d'un fichier.**
  - Syntaxe : head [-num] fichier
  - Option principale : -num : affiche les num premières lignes du fichier. Par défaut, num = 10.
  - Exemple : head -10 *.c : affiche les 10 premières lignes de tous les fichiers sources C du répertoire courant.
- **Tail: affiche la fin d'un fichier.**
  - Syntaxe : tail [-num] fichier
  - Option principale : -num : affiche les num dernières lignes du fichier. Par défaut, num = 10.
  - Exemple : tail -50 essai : affiche les 50 dernières lignes du fichier essai.

Processus: Gestion des processus

- **Tac: affiche le contenu d'un fichier en ordre inverse, c'est à dire en commençant par la fin.**
  - Syntaxe : tac fichier
  - Exemple : tac essai.txt : affiche le fichier essai.txt en ordre inverse, c'est à dire en commençant par la fin.
- **Nl: affiche les lignes d'un fichier précédées de leur numéro.**
  - Syntaxe : nl [-ba] fichier
  - Options principales : -ba : numérote aussi les lignes blanches (ou vides).
  - Exemple : nl -ba essai.c : affiche toutes les lignes du fichier essai.c précédées de leur numéro.
- **Wc: compte le nombre de lignes, de mots ou de caractères d'un fichier.**
  - Syntaxe : wc [-clw] fichier
  - Options principales : -c : renvoie le nombre de caractères. -l : renvoie le nombre de lignes. -w : renvoie le nombre de mots.
  - Exemple : wc -l essai.c : renvoie le nombre de lignes du fichier essai.c.

Processus: Gestion des processus

- **Uniq: recherche les lignes consécutives identiques dans un fichier.**
  - Syntaxe : uniq [-udc] [+m] [-n] fichier
  - Options principales : -u : affiche les lignes consécutives identiques une seule fois. -d : n'affiche que les lignes consécutives identiques. -c : chaque ligne est précédée de son nombre d'occurences. +m : saute les m premiers caractères de chaque ligne. -n : saute les n premiers champs non blancs. Exemple : Prenons le fichier noms :

    alain
    jean
    maurice
    alain
    robert
    jean

  - Après l'avoir trié en tapant : sort noms -o nomstries , la commande : uniq -c nomstries renvoie

    2 alain
    2 jean
    1 maurice
    1 robert

25

25

Processus: Gestion des processus

- **Grep: recherche une (ou plusieurs) expression(s) dans un fichier.**
  - Syntaxe : grep [-cilnrvE] expression fichier
  - Options principales : -c : affiche le nombre de lignes contenant l'expression. -i : contrairement à la plupart des commandes, l'option -i ne signifie pas interactif mais ignore-case, c'est à dire que grep ne fera pas de différence entre minuscules et majuscules. -l : n'affiche que le nom des fichiers contenant l'expression. -n : affiche les numéros des lignes contenant l'expression. -r : recherche récursivement dans les sous-répertoires du répertoire courant. -v : inverse la recherche, c'est à dire affiche les lignes ne contenant pas l'expression. -w : recherche exactement l'expression et non les mots contenant l'expression. -E ou --extended-regexp : expression régulière étendue qui permet de rechercher plusieurs expressions en même temps.
  - Exemples :

    grep -i -c 'integer' * : affiche le nombre de fois où les expressions integer, Integer ou INTEGER sont rencontrées dans chaque fichier du répertoire courant.

    grep -Eir "fortran|Python" /codes : cherche les chaînes fortran ou Python, sans tenir compte de la casse, dans le répertoire /codes et ses sous-répertoires.

26

26

Processus: Gestion des processus

- **Sort: trie les champs d'un fichier.**
  - Syntaxe : sort [-fnru] [-t car] [+i] [-j] [-o sortie] fichier
  - Options principales :
  - -f : ne tient pas compte des minuscules et majuscules. -n : trie numérique et non alphabétique. -r : inverse l'ordre de tri. -u : n'affiche qu'une fois les lignes multiples. -t car : indique le caractère séparateur de champs. (Par défaut, c'est l'espace). -o sortie : écrit le résultat dans le fichier sortie. +i : on trie à partir du i-ième champ. --j : on trie jusqu'au j-ième champ. Le premier champ est noté 0, le second 1 ,...
  - Exemples : **sort +1 liste : trie le fichier liste par ordre alphabétique des noms.**
    Alex TERIEUR
    Alain VERSE
    Jean NEMARD
    Robert BIDOCHON
    sort -n -t : +1 -2 adresses : trie le fichier adresses par numéros de téléphones.
    alain:0298123456:Quimper
    alex:0466789012:Marseille
    jean:0144567890:Paris
    robert:0380234567:Dijon

27

27

Processus: Gestion des processus

- **Cut: extrait des champs d'un fichier.**
  - Syntaxe : cut [-cf] [-d car] [+i] [-j] fichier
  - Options principales : -c : extrait suivant le nombre de caractères. -f : extrait suivant le nombre de champs. -d car : indique le caractère séparateur de champs. (Par défaut, c'est la tabulation). +i : on trie à partir du i-ième champ. +j : on trie jusqu'au j-ième champ. Contrairement à sort, le premier champ est noté 1, le second 2 ,...
  - Exemple: **cut -d : -f2 adresses : extrait le deuxième champ**
- **Paste: ajoute des champs à un fichier**

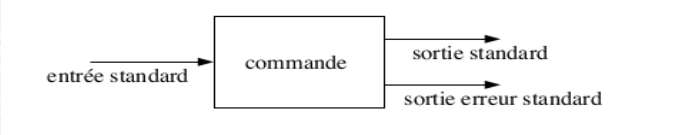
  Syntaxe : paste [-s] [-d car] fichier1 fichier2
  - Options principales : -s : les lignes sont remplacées par des colonnes. -d car : indique le caractère séparateur de champs. (Par défaut, c'est la tabulation).
  - Exemple : considérons le fichier professions : **paste -d : adresses professions > carnet**
  - boucher
  - journaliste
  - instituteur
  - mécanicien

28

28

Processus: Gestion des processus

- Les entrées/sorties
  - Une commande est une "boîte noire" à laquelle on fournit des arguments sur un fichier logique nommé entrée standard et qui renvoie deux types de réponses éventuelles sur des fichiers logiques appelés respectivement sortie standard et sortie erreur standard.



  - Par convention
    - **0 entrée standard** : si le processus a besoin de demander des informations à l'utilisateur, il les lit sur l'entrée standard
    - **1 sortie standard** : si le processus a besoin d'afficher un résultat, il l'écrit sur la sortie standard
    - **2 sorties erreur standard** : si le processus a besoin d'afficher un message d'erreur, il l'écrit sur la sortie erreur standard

29

29

Processus: Gestion des processus

- Rediriger les entrées/sorties
  - En shell, on peut modifier les fichiers identifiés par les descripteurs
    - Rediriger l'entrée standard avec le caractère <. La commande lit ses données dans le fichier indiqué

```
commande < fichier
```

    - Rediriger la sortie standard avec le caractère >. La commande écrit ses résultats dans le fichier indiqué: S'il n'existe pas, il est créé, sinon son contenu est écrasé

```
commande > fichier
```

    - Rediriger la sortie standard avec les caractères >>. La commande écrit ses résultats à la fin du fichier indiqué: S'il n'existe pas, il est créé, sinon les résultats sont ajoutés concaténés en fin de fichier

```
commande >> fichier
```

30

30

Processus: Gestion des processus

- Rediriger les entrées/sorties (récapitulatif)

`< nom_fichier`	utilise `nom_fichier` comme entrée standard
`> nom_fichier`	utilise `nom_fichier` comme sortie standard
`>! nom_fichier`	même si `nom_fichier` existe et **noclobber**
`>& nom_fichier`	inclut les messages d'erreur à `nom_fichier`
`>&! nom_fichier`	même si `nom_fichier` existe et **noclobber**

`>> nom_fichier`
`>>& nom_fichier`
`>>! nom_fichier`
`>>&! nom_fichier`
	même chose avec concaténation en fin de `nom_fichier`

- Exemple
  - ls > res
  - cat > exemple
  - cat < exemple >> res

- Pour activer ou désactiver les options du shell
  - **set -o option** ou **set +o option**

```
saad@saad:~$ set -o
allexport       off
braceexpand     on
emacs           on
errexit         off
errtrace        off
functrace       off
hashall         on
histexpand      on
history         on
ignoreeof       off
interactive-comments    on
keyword         off
```

31

31

Processus: Gestion des processus

- Exemple:
  - La commande *cat* écrit sur la sortie standard ce qu'elle reçoit sur l'entrée standard.
  - La commande *date* ne prend rien sur le flot d'entrée et renvoie quelque chose sur le flot de sortie.
  - La commande *cd* ne prend rien en entrée et ne renvoie rien en sortie.
  - Toutes ces commandes peuvent évidemment renvoyer quelques choses sur la sortie erreur standard pour signaler un problème quelconque (mauvaise utilisation de la commande, arguments inexistants, problèmes de droits, etc.).
  - Par commodité, il est souvent nécessaire de sauver les données fournies par un programme dans un fichier pour pouvoir ensuite les voir en totalité ou les traiter ou inversement de réutiliser des données qui sont déjà dans un fichier.
  - Par défaut, le canal d'entrée est le clavier et les canaux de sortie et sortie erreur sont l'écran.

32

32

Processus: Gestion des processus

- Exemple 1:
  - Tapez *ps* dans une fenêtre de terminal. Tapez ensuite *ps* > *fic*. Un nouveau fichier nommé *fic* a été créé. Regardez son contenu.
  - Maintenant faites *date* > *fic*. Regardez à nouveau le contenu du fichier *fic*. Concluez sur le rôle de >
  - Faites *who* >> *fic*. Regardez à nouveau le contenu du fichier *fic*. Concluez sur le rôle de >>.

33

33

Processus: Gestion des processus

- Exemple 2
  - Lancez *cat* sans argument. Tapez quelques caractères puis frappez la touche entrée. Observez le résultat.
  - Tapez encore quelques lignes, puis pour indiquer au processus la fin des données à traiter, pressez la combinaison de touches *Ctrl-D* (end of file).
  - Que se passe-t-il si à la fin d'une ligne contenant des caractères, vous pressez *Ctrl-C* au lieu de *Ctrl-D* ? Expliquez.
  - Lancez maintenant la commande *cat* < *fic*. Comment expliquez-vous ce qui se passe.

34

34

Processus: Gestion des processus

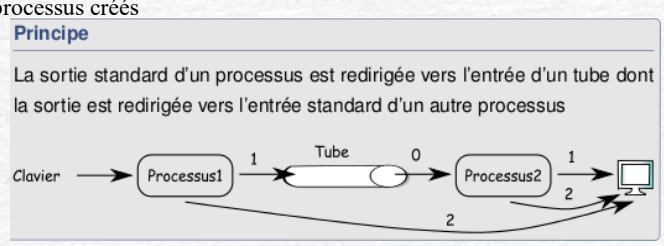
- Exemple 3:
  - Tapez dans votre terminal la commande dater (qui n'existe pas) : vous obtenez un message d'erreur. Rediriger la sortie standard de cette commande vers un fichier. Regardez le contenu de ce fichier. Que constatez-vous ?
  - En fait, les messages d'erreur ne sortent pas sur le même canal que le résultat des commandes.
  - Recommencez l'opération en redirigeant avec le symbole 2>. Faites des tests pour comprendre la signification du symbole 2>>.

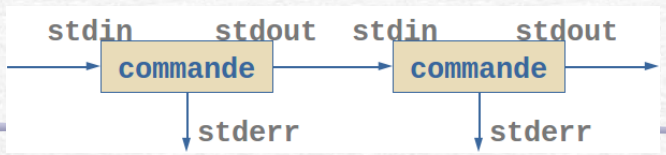
35

35

Processus: Gestion des processus

- Communication entre processus : les tubes
  - Plusieurs processus qui s'exécutent en parallèle peuvent communiquer entre eux
  - Il est possible de réaliser cette communication par l'intermédiaire de tubes (pipes)
  - Le système est alors chargé d'assurer la synchronisation de l'ensemble des processus créés





36

36

Processus: Gestion des processus

- Les tubes : utilisation
  - Le lancement en parallèle des processus communiquant par un tube est réalisé grâce au caractère pipe. La sortie de *commande1* est l'entrée de *commande2*

```
commande1 | commande2
```

  - On peut utiliser plusieurs tubes :

```
Commande 1 | commande 2 | ... | commande n
```

```
saad@saad:~$ gedit &
[1] 3535
saad@saad:~$ ps aux | grep gedit
saad        3535  2.3  0.7 560184 58428 pts/0    Sl   09:38   0:00 gedit
saad        3545  0.0  0.0   9144  2268 pts/0    S+   09:38   0:00 grep --color=auto ged
it
```

37

37

Processus: Gestion des processus

- **Exemple: Redirections, méta-caractères, find, head, tail, tr, more, grep, cut, avec tubes**
  - Le répertoire /usr/include contient les fichiers d'entête standards en langage C
  - Créer un répertoire nommé inc dans votre répertoire de connexion (HOME). En utilisant une seule commande, y copier les fichiers du répertoire /usr/include dont le nom commence par std.

    mkdir inc; cp /usr/include/std* $HOME/inc

  - Afficher la liste des fichiers de /usr/include dont le nom commence par a, b ou c.

    find /usr/include/ \(-name 'a*' -o -name 'b*' -o -name 'c*' \) ou

    find /usr/include/ -name '[abc]*'

  - Modifier la commande de la question précédente pour qu'au lieu d'afficher le résultat, celui-ci soit placé dans un fichier nommé "Abc.list" de votre répertoire de connexion

    find /usr/include/ \(-name 'a*' -o -name 'b*' -o -name 'c*' \) >$HOME/Abc.list

38

38

Processus: Gestion des processus

- Exemple: Redirections, méta-caractères, find, head, tail, tr, more, grep, cut, avec tubes
  - Afficher la liste des fichiers h situés sous le répertoire /usr/include.

    find /usr/include -name "*.h"
  - Afficher la liste des fichiers plus vieux que 3 jours situés sous votre répertoire de connexion

    find $HOME -ctime +3
  - Afficher les 5 premières, puis les 5 dernières lignes du fichier /etc/passwd

    head -5 /etc/passwd; tail -5 /etc/passwd
  - Afficher la 7ième ligne de ce fichier (et elle seule), en une seule ligne de commande

    head -n7 passwd | tail -n1
  - Afficher le fichier /etc/passwd en remplaçant les caractères / par des X

    cat /etc/passwd | tr / X
  - Obtenir le résultat précédent page par page

    cat /etc/passwd | tr / X | more

39

39

Processus: Gestion des processus

- Exemple: Redirections, méta-caractères, find, head, tail, tr, more, grep, cut, avec tubes
  - Afficher le contenu de ce fichier en utilisant la commande cat. Copier avec cat son contenu dans un nouveau fichier nommé "Copie".

    cat Abc.list > copie; wc Abc.list
  - Toujours avec cat, créer un nouveau fichier nommé "Double" formé par la mise bout à bout (concaténation) des fichiers "Abc.list" et "Copie". Vérifier que le nombre de lignes a bien doublé à l'aide de la commande wc.

    cat Abc.list copie >double; wc double
  - Créer un fichier nommé "Temp" contenant une ligne de texte. Avec cat, ajouter une ligne "The end" à la fin du fichier "temp".

    touch Temp
    cat >> temp The end CTRL-d
  - En une seule ligne, faire afficher le nombre de fichiers de /usr/include dont le nom contient la lettre t.

    ls /usr/include/[t]* | wc

40

40

Processus: Gestion des processus

- Exercice
  - On dispose d'un fichier texte telephone.txt contenant un petit carnet d'adresses. Chaque ligne est de la forme "nom prenom numerotelephone" les champs étant séparés par des tabulations. Répondre aux questions suivantes en utilisant à chaque fois une ligne de commande shell:

Durand	Emilie	0381818585
Terieur	Alex	0478858689
Tinrieur	Georges	0563868985
Dupond	Albert	04961868957
Dupont	Emilie	02971457895
Dupond	Albertine	0131986258
Bouvier	Jacques	0381698759
Zeblues	Agathe	0685987456
Dupond	Agnès	0687598614
Dumont	Patrick	04661645987
Dupond	Elisabeth	0654896325
Houtand	Laurent	0658769458

- Afficher le carnet d'adresse trié par ordre alphabétique de noms
- Afficher le nombre de personnes dans le répertoire
- Afficher toutes les lignes concernant les "Dupond"
- Afficher toutes les lignes ne concernant pas les "Dupond"
- Afficher le numéro de téléphone (sans le nom) du premier "Dupond" apparaissant dans le répertoire
- Afficher le numéro de téléphone (sans le nom) du premier "Dupond" dans l'ordre alphabétique (ordre basé sur les prénoms).

41

41

Processus: Gestion des processus

- Afficher le carnet d'adresse trié par ordre alphabétique de noms

  sort telephone.txt

- Afficher le nombre de personnes dans le répertoire

  wc -l telephone.txt

- Afficher toutes les lignes concernant les "Dupond"

  grep Dupond telephone.txt

- Afficher toutes les lignes ne concernant pas les "Dupond"

  cat telephone.txt | grep -v "Dupond"

- Afficher le numéro de téléphone (sans le nom) du premier "Dupond" apparaissant dans le répertoire

  grep Dupond telephone.txt|uniq -u|cut -f 3

- Afficher le numéro de téléphone (sans le nom) du premier "Dupond" dans l'ordre alphabétique (ordre basé sur les prénoms).

  sort telephone.txt|grep Dupond |uniq -u|cut -f 3

42

42

Processus: Gestion des processus

- Contrôler l'exécution d'un processus: les signaux
  - Les programmes ne fonctionnent malheureusement pas toujours comme prévu. Un garant important de la stabilité du système est donc de pouvoir mettre fin à l'exécution de processus devenus instables ou ne répondant plus. Il existe plusieurs méthodes pour mettre fin à des processus récalcitrants.

Qu'est-ce qu'un signal ?

- Un signal est un message envoyé par le noyau à un processus pour lui indiquer l'occurrence d'un événement dans le système
- L'information réside dans le nom du signal
- Le processus prend en compte le signal en exécutant une fonction de gestion du signal

Comment un processus reçoit-il un signal ?

- L'exécution d'un processus a déclenché une exception que le gestionnaire d'exceptions associé va indiquer en émettant un signal :
  - Exemple : un processus qui effectue une division par 0 reçoit le signal `SIGFPE`
- Un signal est envoyé par un autre processus à l'aide de la primitive `kill()`
  - Le shell envoie des signaux grâce à la commande `kill`

43

43

Processus: Gestion des processus

- Les signaux

Quelques signaux

- `SIGINT` (2) : Interruption du clavier (*Ctrl-C*)
- `SIGKILL` (9) : Terminaison forcée du processus
- `SIGCONT` (18) : Reprise de l'exécution d'un processus stoppé
- `SIGSTOP` (19) : Stoppe l'exécution d'un processus
- `SIGSTP` (21) : Demande d'arrêt du processus depuis le clavier (*Ctrl-Z*)

La commande kill du csh

```
kill -NUMSIGNAL pid
```

- Envoie le signal de numéro `NUMSIGNAL` au processus dont le pid est `pid`

➥ Par exemple

```
kill -9 2312
```

- Le shell envoie le signal `SIGKILL` au processus de pid `2312`, qui se termine donc (de manière forcée)

44

44

References

- Luigi Logrippo, **Ordonnancement Processus, slides.** http://w3.uqo.ca/luigi/
- http://perso.univ-rennes1.fr/pascal.gentil/docs/unix/unix.fichiers.html#cat1
- http://upsilon.cc/~zack/teaching/1314/progsyst/#index1h2
- Karim Sehaba, Systèmes et Architectures, http://perso.univ-lr.fr/ksehaba/
- Vincent Granet, Introduction à Linux, Polytech'Nice-Sophia
- Sylvain CHERRIER, Cours Linux,
- Gestion des processus, Telecom-ParisTech BCI Informatique