Unix : shells et gestion de processus

Viviane Gal
CNAM – Département Informatique/Laboratoire CEDRIC

bibliographie

- Les systèmes d ’exploitation

  Unix, Linux et Windows XP avec C et Java

  Cours et exercices corrigés

  Samia Bouzefrane - CNAM

  Editions Dunod
- UNIX

  Programmation et communication

  Jean-Marie Rifflet, JeanBaptiste Yunès

  Editions Dunod
- Cours Unix

  http://www.linux-France.org/article/dalox/
- Cours C-shell « Le langage de commande du système Unix »

  Gérard Florin - CNAM

Unix : shells et gestion de processus

- Introduction à une prise en main
- Mécanismes généraux
- Langages de commandes / interpréteurs
- Gestion de processus

Introduction à une prise en main

Unix : shells et gestion de processus

- Session de travail
  - Session en mode texte
    - login :
    - passwd :
    - $, % ou autres
    - Terminaison interpréteur
      - <CTRL>-D
      - exit ou logout
  - Session en mode graphique
    - Terminaux modernes à capacités graphiques
    - Terminal X ou console graphique = équipement terminal
    - Connexion fonction configuration logicielle
    - Procédure d'identification
    - Session de travail X
    - xterm

Introduction à une prise en main

Unix : shells et gestion de processus

Quelques commandes :

- La forme des commandes

  2 classes : externes et internes

  commande [argument$_1$ … argument$_n$]

- Le fonctionnement interactif d'un shell

- Des commandes élémentaires
  - echo
  - date
  - who
  - passwd

- La réaction aux commandes incorrectes

  Message adapté à l'erreur

- La terminaison d'un shell

Exercices

- echo

```
kirov> echo commande1 ; pwd ; echo commande2
commande1
/users/ensinf/viviane
commande2
```

- date

```
kirov> date
mar oct 26 13:51:39 CEST 2004
```

- who

```
kirov> who
irazac   pts/0        Oct 26 09:54 (neoph1.cnam.fr)
tartar_m pts/2        Oct 26 10:29 (tx5-04.cnam.fr)
tartar_m pts/1        Oct 26 12:26
viviane  pts/3        Oct 26 12:58 (eoliane.cnam.fr)
elhach_f pts/4        Oct 26 13:05 (tx5-08.cnam.fr)
```

- id

```
kirov> id
uid=2423(viviane) gid=23(squid) groupes=23(squid)
```

- stty -a

```
kirov> stty -a
speed 9600 baud; rows 28; columns 112; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
   eol2 = <undef>; start = ^Q; stop = ^S;
…
```

Mécanismes généraux

Unix : shells et gestion de processus

- Qu'est-ce qu'un processus ?
  - Entité dynamique = reflet état d'exécution d'un programme
  - Création par duplication dans Unix
  - Un parent ou père => arborescence

Mécanismes généraux (suite 1)

Unix : shells et gestion de processus

- Processus
  - Programme en cours d'exécution
  - Ensemble de données
  - Ensemble d'informations (contexte d'exécution)
    - Ensemble valeurs registres du processeur
    - Etat du processus
    - Informations liant le processus avec l'extérieur

} Bloc de contrôle du processus

Mécanismes généraux (suite 2)

Unix : shells et gestion de processus

- Ordonnancement des processus

  Activités dans système exécutées dans contexte processus

  Un rôle du système = permettre aux processus de s'exécuter

  - Ordonnanceur alloue le ou les processeurs aux processus
  - Processus = objet de base pour ordonnanceur
  - 2 classes d'ordonnancement
    - Partage de temps avec attribution dynamique et stratégie du tourniquet (classe TS)
    - Temps réel avec comme critère le temps de réponse (classe RT)

Mécanismes généraux (suite 3)

Unix : shells et gestion de processus

- Types de processus
  - Processus système
    - Processus démon (daemon)
    - Propriétaire : utilisateur privilégié
    - Rôle : assure services généraux accessibles
    - Situation : racine absolue
    - Création : au lancement ou à dates fixées
    - Exemples : initd, lpsched ou crond
  - Processus utilisateur
    - Dédiés à l'exécution de tâches particulières à durée de vie limitée mais non bornée
    - Lancés par les utilisateurs

Mécanismes généraux (suite 4)

Unix : shells et gestion de processus

- Exécution des processus

  - Nécessite un programme binaire
  - Sous Unix programmes réentrants
  - Pour les données = espace d'adressage du processus.
    - Différentes approches :
      - Propre jeu de données (système unix classique)
      - Partage de mêmes suites d'instructions et aussi de mêmes données physiques. Concept de threads par la norme posix.

Mécanismes généraux (suite 5)

Unix : shells et gestion de processus

- Caractéristiques d'un processus

  Certains de ces éléments peuvent changer au cours du temps

  – Son identification (un nombre entier)
  – L'identification du processus parent ou père : uid
  – Ses propriétaires
  – Ses groupes propriétaires : gid
  – Son terminal d'attachement (éventuellement)
  – Des attributs
    – Priorité
    – Répertoire de travail
    – Différents temps
    – …

Mécanismes généraux (suite 6)

Unix : shells et gestion de processus

- La commande ps
  - Permet l'obtention de la liste des processus satisfaisant certains critères + leurs caractéristiques
  - Sans option -> liste processus même session

Mécanismes généraux (suite 7)

Unix : shells et gestion de processus

- Terminaison des processus

  – Etat zombi

    Plus de ressource mais entrée table processus

  – Code retour d'un processus

    - Valeur du processus qui se termine (nul si normal)
    - Le père y accède

  – Interruption d'un processus

    - Directement par la frappe de caractères particuliers => signaux (interruptions ou événements logiciels) de noms symboliques

      Exemples : CTRL C, DEL ou CTRL \

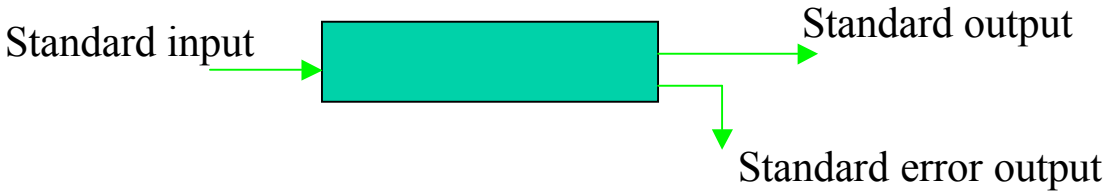
      SIGINT ou SIGQUIT ⇔ entiers 2 et 3

Mécanismes généraux (suite 8)

Unix : shells et gestion de processus

- Fichiers standard et redirections
  - Applications développées réutilisables et composables simplement
  - Applications = boîtes noires
    - Lisent données sur fichier logique = entrée standard du processus
    - Ecrivent résultats sur fichier logique = sortie standard du processus
  - En interne, fichiers logiques => aux 2 1ères entrées dans la table des fichiers manipulés par le processus (descripteurs).

Modification de l'association d'un tel descripteur à un fichier physique
=
Mécanisme de redirection



Mécanismes généraux (suite 9)

Unix : shells et gestion de processus

- Enchaînement de processus

  Il est possible d'enchaîner l 'exécution de deux processus de façon indépendante et en séquence

  – Deux caractéristiques essentielles :
    - Séquentiels
    - Indépendants
  – Procédé rudimentaire, suppression fichiers temporaires
  – Autres mécanismes

Mécanismes généraux (suite 10)

Unix : shells et gestion de processus

- Enchaînement de processus (suite 1)
  - Processus concurrents et communication par tube
    - Mécanisme : en // et communiquant avec système pour la synchronisation
  - Comment ?

$commande_1 \mid commande_2 \mid \ldots \mid commande_n$

Mécanismes généraux (suite 11)

Unix : shells et gestion de processus

Exercice

- Comment se connecter. Savez-vous ce qui se passe à la connexion ?
  - Login
  - Password

Mécanismes généraux (suite 12)

Unix : shells et gestion de processus

Exercice

- Examinez les répertoires :
- /etc/passwd
- /etc/group

Mécanismes généraux (suite 13)

Unix : shells et gestion de processus

Exercices

`ps` sur machine locale

`telnet « machine distante »` demande de connexion distante

`ps` sur machine distante

`CTRL D` pour shell distant

retour au shell local

`ps -l`

Mécanismes généraux (suite 14)

Unix : shells et gestion de processus

Exercices : redirection entrée standard

```
cat fichier
```

```
wc < fichier
```
nbre de lignes, de mots, de caractères lus sur entrée standard

```
wc fichier
```
nbre de lignes, de mots, de caractères lus dans le fichier

Mécanismes généraux (suite 15)

Unix : shells et gestion de processus

Exercices : sortie erreur standard

Créez un fichier « fica » ou « autre nom »

```
ls fica fico
  fico not found
  fica


ls fica fico > ficu
  fico not found

cat ficu
  fica
```

Exercices : Redirection d'une commande composée puis enchaînement de processus

```
(date;who am i;echo fin de commande composee)>toto
cat toto
```

Résultat ?

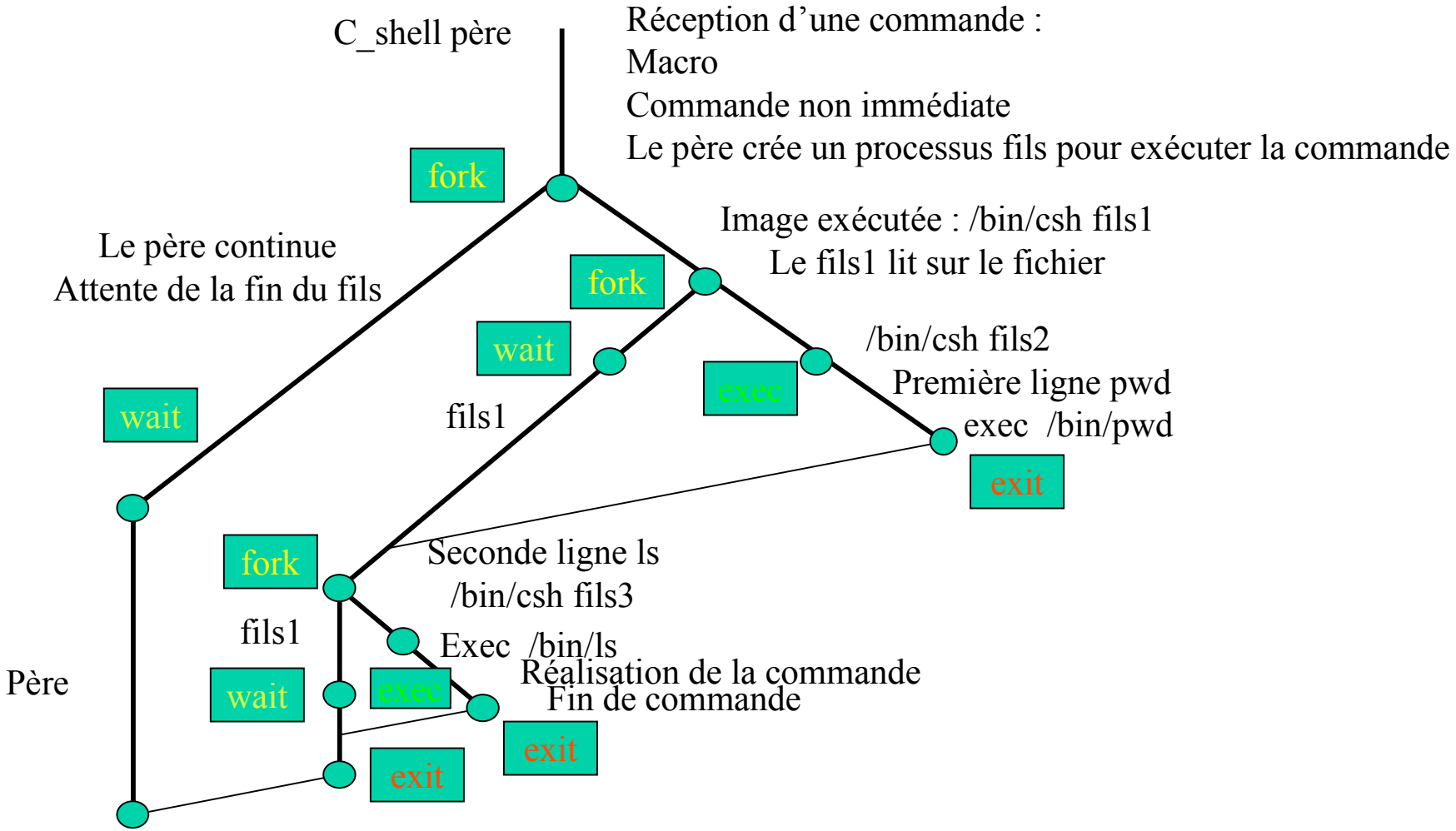
Refaites cette commande sans les parenthèses.

```
ps -e > /usr/tmp/toto
wc -l < /usr/tmp/toto

rm /usr/tmp/toto
ps -e > /usr/tmp/toto ; wc -l < /usr/tmp/toto ; rm /usr/tmp/toto

ps -e | wc -l

ps -l | tee /dev/tty | wc -l
```

Mécanismes généraux (suite 17)

Unix : shells et gestion de processus

Exécution d'une procédure de commande



Langages de commandes - interpréteurs

Unix : shells et gestion de processus

- Avant tout le man

  la documentation en ligne

  - Car on oublie les commandes
  - La syntaxe et la sémantique des options peuvent varier

- Exercice :
  - consultez le manuel
  - man tcsh ou bash puis CTRL Z pour sortir

Langages de commandes - interpréteurs (suite 1)

Unix : shells et gestion de processus

- Différents interprètes de langages de commandes sous Unix : shells

- Shells : double fonction
  - En mode interactif (ligne de commande)
  - En mode non interactif (utilitaires ou scripts) mode batch

- Disposent de :
  - Commandes internes (code fait partie du shell)
  - Commandes externes (commande réalisée par processus dédié)

Langages de commandes - interpréteurs (suite 2)

Unix : shells et gestion de processus

- Deux grandes familles de langages de commandes
  - Bourne-Shells, dérivés du langage de commandes originel d'Unix
  - C-Shells, dérivés du langage C-Shell des distributions BSD

- Même famille = même syntaxe mais fonctionnalités différentes

- Langages de commande = utilitaires associés à des fichiers binaires exécutables

  /bin /usr/bin /usr/ucb ou /usr/local/bin

Langages de commandes - interpréteurs (suite 3)

Unix : shells et gestion de processus

- On trouve les liens :
  - sh — Bourne-Shell
  - rsh — Restricted Shell
  - jsh — Job-Shell
  - ksh — Korn-Shell
  - rksh — Restricted Korn-Shell
  - bash — Bourne-Again-Shell
    (Bourne-Shell étendu du domaine public)
  - csh — C-Shell
  - tcsh — Turbo-C-Shell
  - …

Langages de commandes - interpréteurs (suite 4)

Unix : shells et gestion de processus

- Que se passe-t-il quand un utilisateur se connecte ?

  Un processus correspondant à un interprète Shell est lancé

- L'utilisateur peut vouloir changer de shell

  αshell à βshell

  – Temporairement
    - Si connecté, création du processus βshell en tapant la commande βshell (empilé)
    - Ou en recouvrant le processus αshell par la commande exec βshell (attention, certaines variables risquent de ne plus être définies)
  – Définitivement
    - Soit par l'administrateur
    - Soit en utilisant une des commandes suivantes selon les systèmes

      chsh sous HP/UX, passwd sous Solaris, chpass sous BSD ou nispasswd sou NIS

Le fichier /etc/shells donne la liste des shells autorisés

Langages de commandes - interpréteurs (suite 5)

Unix : shells et gestion de processus

- Caractéristiques générales des différents shells
  - Sémantique particulière (caractères spéciaux ou métacaractères)
  - Différents mécanismes de substitution
    - L'historique
    - L'alias ou le surnom
    - La désignation symbolique
    - La substitution de variables
  - Mécanisme d'expansion
  - Permettent la réalisation de redirection
  - Exécution de commandes en arrrière-plan (background) et en avant-plan (foreground)
  - Communication par tubes
  - Contrôle des différentes tâches (job control)

Langages de commandes - interpréteurs (suite 6)

Unix : shells et gestion de processus

- La ligne de commande
  - On appelle ligne de commande la suite de caractères tapés par l'utilisateur à l'invite du shell et qui se termine par une marque de fin de ligne (retour à la ligne)

- Le bash incorpore des fonctionnalités du C-shell
  - Le nom C-shell vient du langage C (structures de contrôle et expressions), le bash interpréteur de commandes GNU Bourne-Again Shell compatible sh
  - L'invite (prompt) par défaut est % ou $ (csh) et > (bash)

Langages de commandes - interpréteurs (suite 7)

Unix : shells et gestion de processus

- Le C-shell (suite 1)
  - Fichiers de configuration
    - À invocation de l'interprète = exécution dans l'ordre de
      - /etc/csh.cshrc puis .cshrc du répertoire privé de l'utilisateur
    - Dans le cas du login c-shell on a :
      - /etc/csh.cshrc suivi de /etc/csh.login et .cshrc de .login
  - Terminaison dans différentes circonstances (sans parler des terminaisons anormales)
    - Frappe du caractère CTRL D
    - Commandes exit, login ou logout (terminaison du c-shell)

Langages de commandes - interpréteurs (suite 8)

Unix : shells et gestion de processus

- En bash (suite 2)
  - Variables

C-shell autorise la manipulation de tableaux et l'évaluation d'expressions arithmétiques

- Mécanismes de substitution : il faut mettre des back-quotes `
  - Exercice

    echo « listefich = `ls` »

  - Autre exercice

    a=« toto »
    x1=$a
    x2=« a »
    echo $x1
    echo $x2

- Commandes de manipulation set et setenv

  définissent la variable associée à la variable c-shell ou bash (cde set) ou à la variable d'environnement c-shell ou bash (setenv ou env)

Langages de commandes - interpréteurs (suite 9)

Unix : shells et gestion de processus

- Le C-shell (suite 3)
  - Variables prédéfinies
    - Variables utilisées ou modifiées par le shell
      - » argv (liste des arguments du shell)
      - » path (liste des répertoires de recherche des commandes)
      - » prompt (chaîne de caractères utilisée pour l'invite)
      - » status (code de retour de la dernière commande terminée)
    - Variables d'environnement
      - » LOGNAME (nom de l'utilisateur)
      - » …
    - Variables qui peuvent modifier le comportement général de l'interprète
      - » echo (echo des commandes après réalisation de substitution et avant exécution)
      - » history (activation du mécanisme d'historique)
      - » ignoreeof (ignore le CTRL D)

Langages de commandes - interpréteurs (suite 10)

Unix : shells et gestion de processus

- Le C-shell ou bash (suite 4)
  - Les commandes de lancement et de redirections
    - Enchaînement par tube avec l'opérateur |&
      - » Commande |& commande

        Permet d'effectuer les 2 commandes en redirigeant les 2 sorties de la 1ère en entrée du tube et la sortie du tube sur l'entrée standard de la 2nde
    - Les redirections
      - » < référence redirection entrée standard
      - » > référence redirection sortie standard
      - » >! Référence redirection sortie standard avec écrasement du fichier
      - » >> référence redirection sortie standard sans écrasement (concaténation)

Langages de commandes - interpréteurs (suite 11)

Unix : shells et gestion de processus

- En bash (suite 5)
  - Alias
    - Mécanisme d'expansion
    - Définition réalisée par la commande

      alias [nom[valeur]]

      - Exercice

        alias

        alias rm ='rm –i'

        alias

        alias dir='ls -a -l --color'

        alias

        unalias dir

        alias

Langages de commandes - interpréteurs (suite 12)

Unix : shells et gestion de processus

- bash (suite 6)
  - Historique

Exemple : !! Relance la dernière commande

Exercice : se définir un historique de 5 en csh ancienne version

set history = 5

pwd

who

date

history

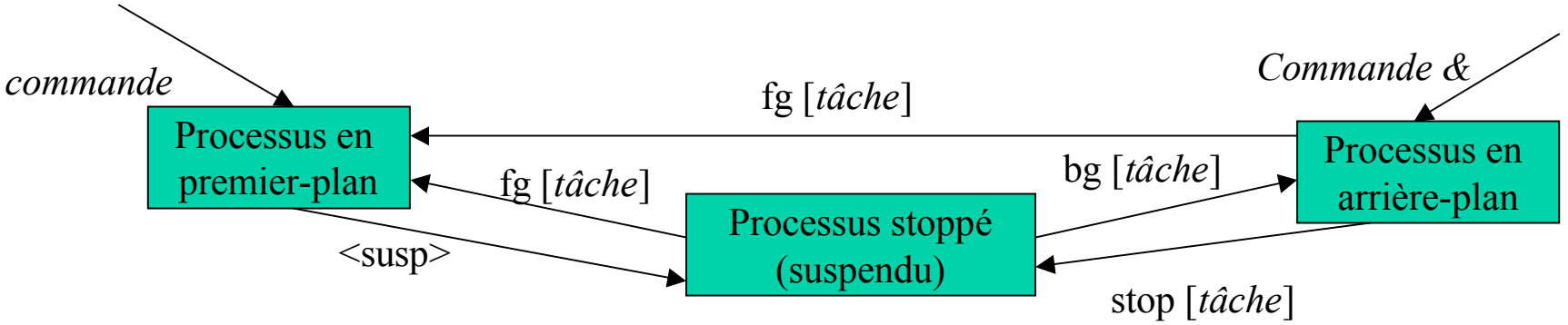
!7

En bash ou nouveau csh, il suffit de faire : history « nombre »

Langages de commandes - interpréteurs (suite 13)

Unix : shells et gestion de processus

- (suite 7)
  - Commandes internes et changements d'état
    - Les commandes internes
      - Jobs [-l] donne la liste des tâches sous la gestion du shell
      - Kill [-signal] % tâche permet d'adresser un signal à une tâche
    - Changements d'état à l'intérieur d'une session



Langages de commandes - interpréteurs (suite 14)

Unix : shells et gestion de processus

- (suite 8)

  – Autres commandes

    - cd [répertoire] — permet de modifier le répertoire courant
    - wait [tâche] — attend la terminaison de n° spécifié ou tous les processus en arrière-plan sans argument
    - …

Gestion de processus

Unix : shells et gestion de processus

- Vu principales caractéristiques
- Visualisons les processus avec les commandes :
  - ps -e
  - ps –l
  - ps -el
- On peut lire
  - UID — identité du propriétaire du processus
  - PID — identité du processus
  - PPID — identité de son père
  - C — informations relatives à la priorité
  - TTY — terminal d'attachement
  - COMMAND — type de processus
  - …

Gestion de processus (suite 1)

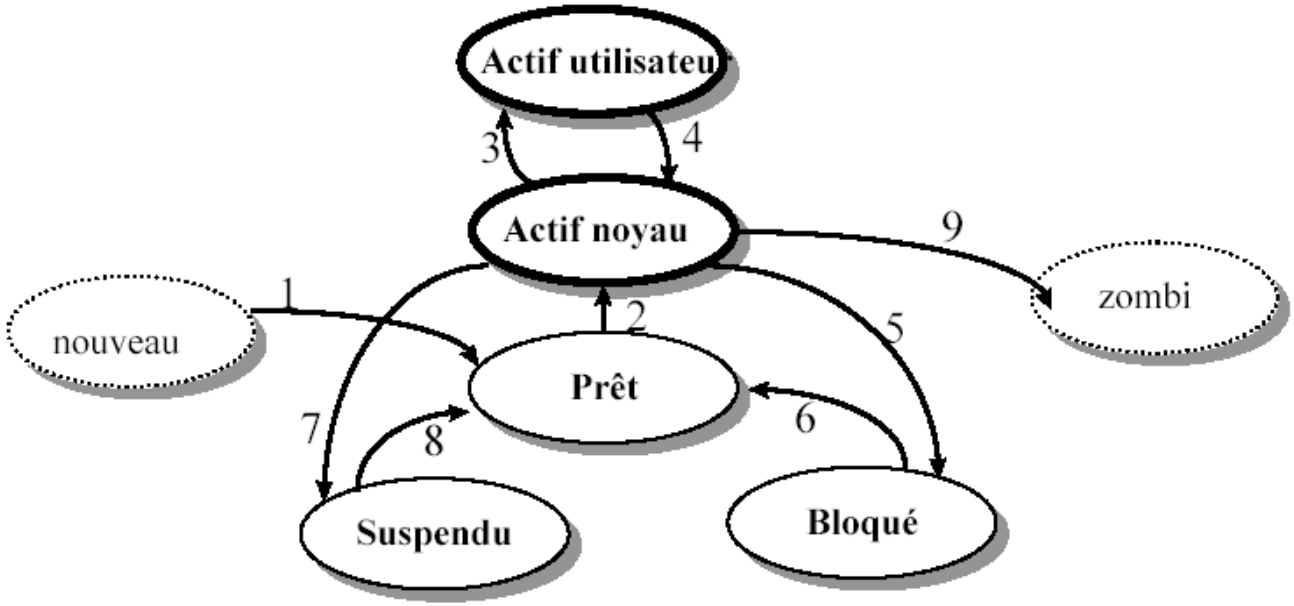
Unix : shells et gestion de processus

- Attributs et primitives générales d'un processus
  - Identité (en général un nombre entier)
  - Propriétaire
    - Réel (renvoyé par un appel de la fonction uid_t getuid(void);)
    - Effectif (accessible par un appel à la fonction uid_t geteuid(void);)
  - Groupe propriétaire
    - Réel
    - effectif
  - Le temps CPU consommé dans les 2 modes utilisateur et noyau est accessible par clock_t times(struct tms *buf);
  - Etat d'un processus (voir schéma page suivante)

Gestion de processus (suite 2)

Unix : shells et gestion de processus

Etat d'un processus Unix



1: allocation de ressources
2: élu par le scheduler
3: le processus revient d'un appel système ou d'une interruption
4: le processus a fait un appel système ou a été interrompu
5: se met en attente d'un événement
6: événement attendu est arrivé
7: suspendu par un signal SIGSTOP
8: réveil par le signal SIGCONT
9: fin

- Création de processus et terminaison
  - Fork primitive de création de processus
    - Fork() crée un processus fils par duplication

```
# include <unistd.h>
pid_t       fork(void);
```

  - Exercice

```
/* fork.c */
#include <unistd.h>
#include <stdio.h>
main() {
pid_t  pid; /* on peut aussi déclarer int pid */
switch (pid=fork())  {
case -1: perror("creation de processus");   exit(2);
case  0:            /* corps du fils */
          printf("je suis le fils %d, de pere %d\n",
getpid(), getppid());
          printf("fin du processus fils\n"); exit(0);
default:  /*  suite du pere  */
          printf("je suis le processus %d de pere %d\n",
getpid(), getppid());
          sleep(1) ;
          printf("fin du processus pere\n");
}
}
```

Gestion de processus (suite 4)

Unix : shells et gestion de processus

- Création de processus et terminaison
  - Exec

    Remplace le code (image binaire) d'un processus en cours d'exécution par une autre code (image binaire) et reprend l'exécution au début du nouveau code.

    On charge en mémoire à la place du processus fils un autre programme provenant d'un fichier binaire.

```
if((pid = fork()) == 0)
  { execl(".....", "...","...",NULL); }
else if (pid>0)
  { suite du code exécuté par le père}
else    { erreur };
```

Gestion de processus (suite 5)

Unix : shells et gestion de processus

- Création de processus et terminaison
  - Exit

    Commande de terminaison d'un processus associé à un code de retour (exit status).

    Par convention la fin normale est 0.
  - Wait

    Suspend l'exécution d'un processus en attente de la terminaison de l'un de ses fils.

```
#include <sys/types.h>
#include <sys/wait.h>
Pid_t    wait(int *etat);
```

Gestion de processus (suite 6)

Unix : shells et gestion de processus

- Création de processus et terminaison
  - Fork primitive de création de processus
    - Fork() crée un processus fils par duplication

```
# include <unistd.h>
pid_t      fork(void);
```

  - Exemple

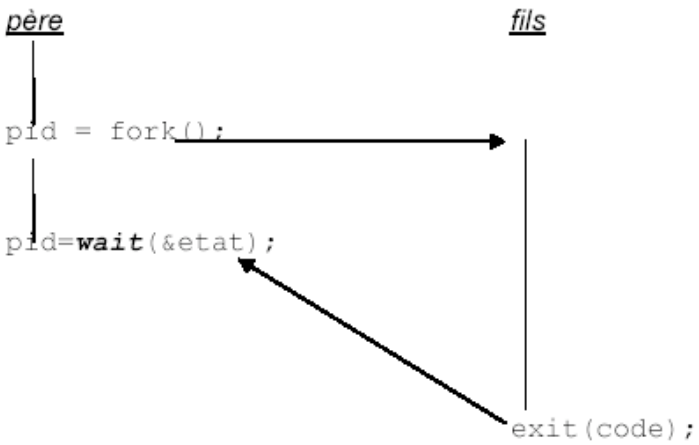
Synchronisation à l'aide de Wait et exit

```
int pid,code, etat;
switch (pid=fork())
{
      case -1:          {perror("erreur dans fork"); exit(1);}
      case 0:     {.................. /* fils */
                   ...................
                   exit(code);
                  }
        default:  { wait(&etat);      /*père*/
                    ...................
                          ...................
                    }
  }
```

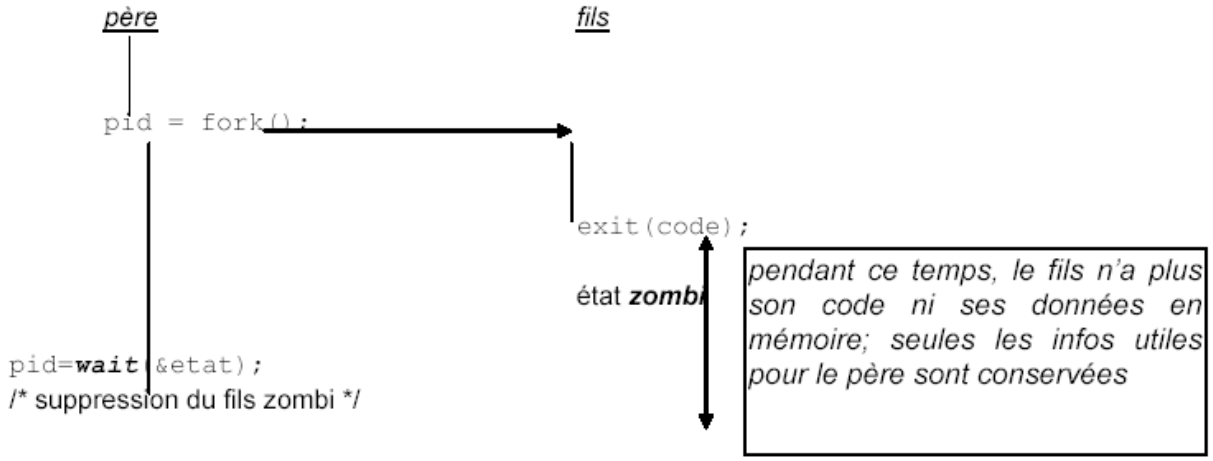
Gestion de processus (suite 7)

Unix : shells et gestion de processus

Utilisation de wait



Processus zombi:



Gestion de processus (suite 8)

Unix : shells et gestion de processus

Une solution : utiliser la primitive waitpid qui est plus évoluée que wait.
Elle permet de tester la terminaison d'un processus particulier en bloquant ou non l'appelant

```
#include <sys/types.h>
#include <sys/wait.h>
pid_t waitpid(pid_t pid, int *etat, int options);
```