

Programmation Système

Stéphanie Moreaud





`stephanie.moreaud@labri.fr`
Département d'informatique
IUT Bordeaux 1

Plan du cours

- 1 Introduction
- 2 Complément de langage C
- 3 Vue générale d'un système d'exploitation
- 4 Généralités sur les processus UNIX
- 5 Gestion des processus sous UNIX
- 6 Communication par signaux
- 7 Communication des processus par tubes
- 8 Mémoire partagée
- 9 Processus légers (threads)

- 1 Généralités sur les processus UNIX
 - Processus UNIX
 - Espace mémoire d'un processus UNIX
 - pid et filiation
 - Fonction `system()`

Bibliographie

-  RIFFLET (J.-M.) et YUNÈS (J.-B.), *UNIX Programmation et communication*. Dunod, 2003.
-  BILLAUD (M.), *Programmation système et réseau*.
Polycopié de cours,
<http://www.labri.fr/perso/billaud/>.
-  NAMYST (R.), *Cours de programmation système*.
-  BILLAUD (M.) et PIERRE (R.), *Supports de cours et TDs*.

Remerciements à Pierre Ramet, Michel Billaud et Kristian Kocher

Plan

- 1 Généralités sur les processus UNIX
 - Processus UNIX
 - Espace mémoire d'un processus UNIX
 - pid et filiation
 - Fonction `system()`

Processus UNIX

Instance de programme en cours d'exécution.

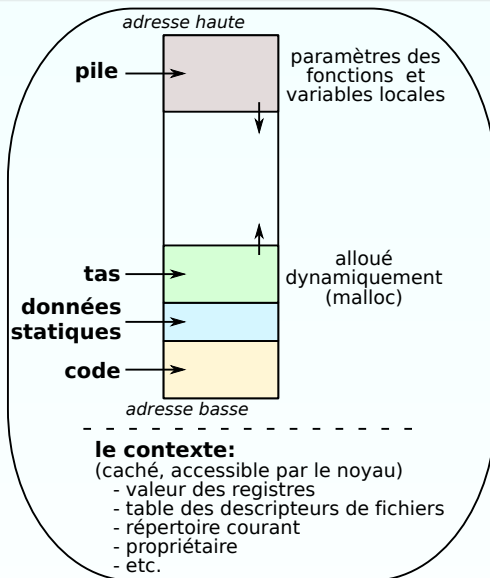
Possède un certain nombre de propriétés/ressources :

- un espace d'adressage privé (adresses virtuelles)
- un numéro d'identification, pid (*Process Identifier*)
- un numéro de propriétaire (propriétaire de l'exécutable)
- un numéro d'utilisateur
- une table de descripteur des fichiers
- le pid du processus parent (père)

Plan

- 1 Généralités sur les processus UNIX
 - Processus UNIX
 - Espace mémoire d'un processus UNIX
 - pid et filiation
 - Fonction `system()`

Espace mémoire d'un processus UNIX



Plan

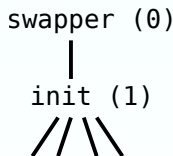
- 1 Généralités sur les processus UNIX
 - Processus UNIX
 - Espace mémoire d'un processus UNIX
 - pid et filiation
 - Fonction `system()`

pid et filiation

A la création chaque processus reçoit un numéro d'identification unique (entier positif), le **pid**.

Un processus est créé par un autre processus

- exception : le processus **swapper** de **pid 0** est créé artificiellement au chargement du système
 - **swapper** crée ensuite le processus **init** de **pid 1**, qui initialise le temps-partagé, et crée des processus *fils*
- l'ensemble des processus existants à un instant donné forme un arbre de parenté dont la racine est le processus initial **init**.



pid et filiation

Les données relatives à chaque processus inclues son pid et le pid de son père.

Elles sont accessibles par l'intermédiaire de fonctions système :

- `getpid()` renvoie le pid du processus
- `getppid()` renvoie le pid du père

Exemple : `identite.c`

```
1  #include <stdlib.h>
2  #include <unistd.h>
3  #include <stdio.h>
4
5  int main(){
6      printf("Je suis un programme en cours d'exécution\n");
7      printf("Je suis le processus n°%i\n", getpid());
8      printf("Le pid de mon père est %i\n", getppid());
9      return 0;
10 }
```

Plan

- 1 Généralités sur les processus UNIX
 - Processus UNIX
 - Espace mémoire d'un processus UNIX
 - pid et filiation
 - Fonction `system()`

Fonction system()

La fonction system() permet de lancer l'exécution d'une commande shell

`system()`

```
#include <stdlib.h>
```

```
int system(const char* command);
```

Code de retour

- 0 si OK, -1 sinon

Fonction system()

Exemple : dater.c

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main() {
5     printf("Nous sommes le : ");
6     fflush(stdout);
7     system("/bin/date +%D");
8     return 0;
9 }
```

Fonction system()

Interprétation : l'exécutable `date` appelle la fonction `system()` qui lance l'exécution d'un processus **shell** interprétant la commande passée en argument

- `system()` crée un nouveau processus qui se termine avec la fin de la commande
- le processus à l'origine de l'appel de `system()` est suspendu jusqu'à la fin de la commande.

```
    dater
    |
sh (bash)
    |
/bin/date
```

Fonction `system()`

Exercice :

- La commande `ps -l` donne une liste des processus courants avec un ensemble étendu d'informations (pour plus de renseignements : `man ps`).
- Dans un terminal, lancez cette commande et constatez les informations listées.
- Depuis le programme `identite.c`, ajoutez l'exécution de cette commande.
- Comparez le résultat avec les messages affichés.