IUT de Montpellier - Programmation Système

Les processus

6 septembre 2022

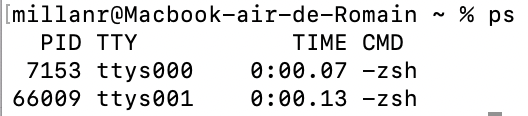
1 Manipulation des processus en Shell (rappels)

Connectez vous sous Linux et ouvrez un terminal.

ps permet de lister les processus en cours

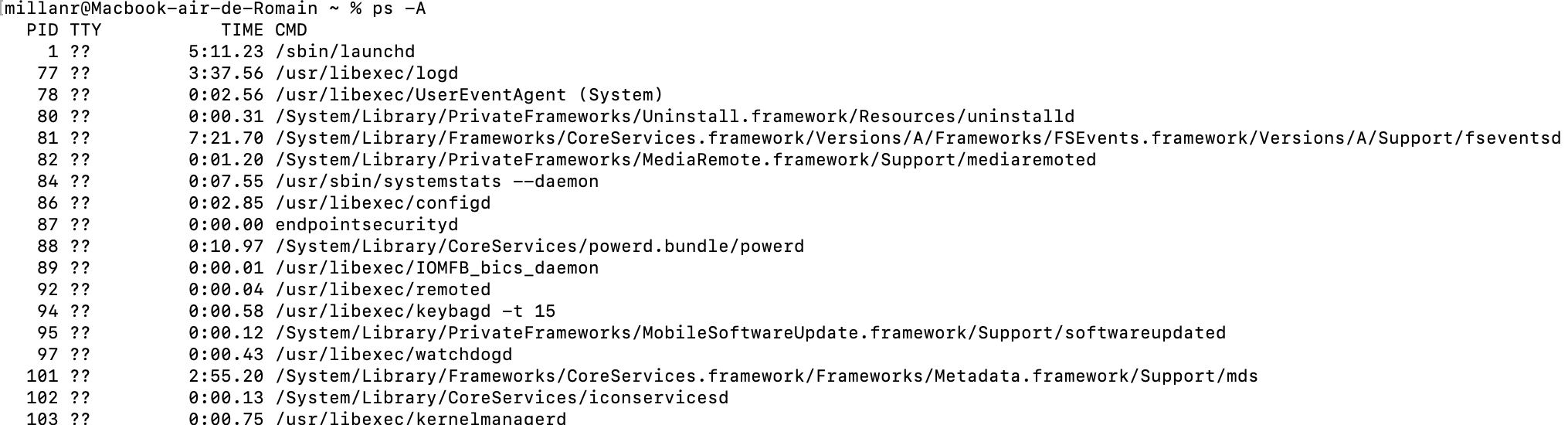
— Lister les processus en cours dans votre terminal. Quel est le PID de votre bash ?

**ps**



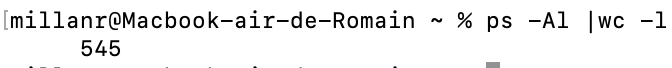
— A l’aide de la page man de ps, trouvez l’option qui permet de lister tous les processus de votre machine. Remarquez que la commande ps est elle-même un processus.

**ps -A**



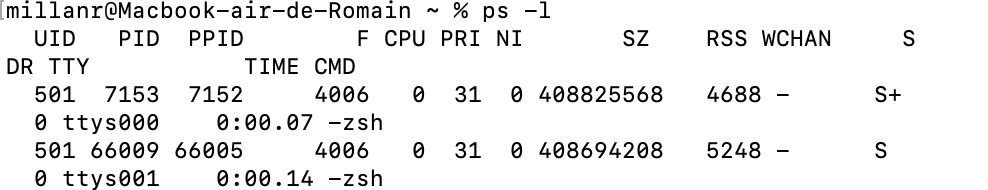
— (*?*) A l’aide d’un pipe et de la commande wc, comptez ce nombre total de processus.

**ps -A | wc -l**



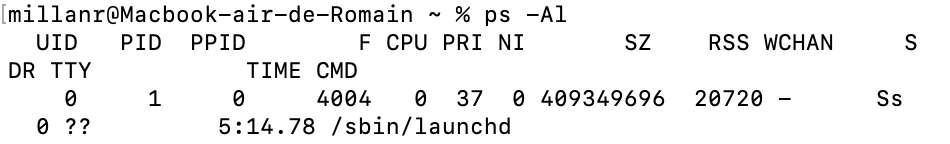
— L'option -l (comme souvent avec les commandes shell) permet d’obtenir plus d’informations. Quel est le PPID de votre bash ?

**ps -l**



— Quel est le processus parent du processus init (ou /sbin/launchd sur un Mac) ?

**ps -Al | head -2**



— (*?*) De quelle génération est votre processus bash (c.a.d. combien de parents lui et le processus init) ? Vous pouvez vous aider d’un pipe et de la commande grep pour ne pas avoir a lire tous les processus

Etape 1: on récupère le PPID du processus

**ps -l**

Étape 2: on itère avec le PPID retourné jusqu'à atteindre le processus de PID 1.

**ps -Al | grep pid\\_du\\_ppid précédent**

— lancer la commande sleep pour 10 secondes

**sleep 10**

— lancer la commande sleep pour 10 secondes en arrière-plan. Quel est le père du processus sleep.

**sleep 10 & ps -l**

2 Programmation C

Pour cette partie du TD reportez vous à la section Le PID du support de cours fourni en Amphi.

2.1 Compiler un programme C (rappel)

— Créez un fichier hello.c et insérez y le code suivant :

1 #include *<*stdio.h*>*

2 i n t main ( v oid ) *{*

3 p r i n t f ( ” Hello World ! ” ) ;

4 *}*

- Compilez ce fichier pour obtenir un exécutable nommé hello. Exécutez le

2.2 Gestion des arguments

Compilez et testez ce petit programme simple de manipulation des arguments.

1 #include *<*s t d i o . h*>*

2 int main (int argc , char ∗argv [ ]) *{*

3 int i =0;

4

5 printf ( ”*\*n Le nombre d ’ arguments est =%d” , a r g c ) ;

6

7 /∗ Le premier argument e s t l e nom du programme ∗/

8 printf ( ”*\*n Cet exécutable est =%s ” , argv [ 0 ] ) ;

9

10 for ( i =1; i*<* a r g c ; i++) *{*

11 printf ( ”*\*n Argument %d=%s ” , i , argv [ i ] ) ;

12 *}*

13 printf ( ”*\*n” ) ;

14 *}*

2.3 Affichage du pid

Ecrire un programme C afichePid qui

1. affiche son pid

2. s’endort pour 10 secondes grâce `à l'appel sleep

1 #include *<*unistd . h*>*

2 unsigned int slip ( unsigned int seconds) ;

Remarque Ici on donne la signature de la méthode. Dans votre code C, il faut juste ajouter l’instruction sleep(10)

3. fini proprement `à l'aide de l’appel exit(0).

1 #include *<*stdlib.h*>*

2 void exit (int status);

Rappel sur les codes de retour d’un programme

Il n’y a pas d’exception en C (comme en JAVA). Toute commande LINUX a un code de retour. Si celui-ci vaut 0, c’est que la commande s’est bien passé. Sinon elle renvoie un entier (diff ́erente de 0). En shell, on r´ecup`ere le code de retour de la derni`ere commande ex ́ecut ́ee avec echo $?

bash-3.2$ ls public\_html

index.html

bash-3.2$ echo $?

0

bash-3.2$ ls public\_html/truc.php. // un fichier qui n’existe pas

ls: public\_html/truc.php: No such file or directory

bash-3.2$ echo $?

1. //diff´erent de 0, donc il y a eu un problème.

2.4 Affichage du PPID d’un processus

- Ecrire un programme C affichePPid qui prend un argument de de type entier, qui affiche le PPID du processus dont le PID est celui pass´e en param`être du programme.

Pour convertir une chaîne de caractères (ici argv[1] en un entier, vous utiliserez la fonction atoi, dont voici la signature.

1 #include *<*stdlib . h*>*

2 int atoi ( const char ∗str ) ;

— Gestion des erreurs. Faites en sorte que votre programme ai comme code de retour

0 si le PID correspond à un processus existant.

1 si le programme est appelée sans argument

2 si l’argument pass´e n’est pas le PID d’un processus

— (*?*) Écrire un programme C generationPid qui prend comme argument un PID, qui calcule le génération (profondeur dans l’arbre des processus) du processus dont le PID est passée en param`etre du programme et l’affiche. Vous pourrez l'utiliser au choix avec une fonction r´ecursive ou un boucle until. Implémentez les codes de retours du programme affichePPid de la question précédente.