A. Benoit, A.Isoard, A.Lefray et D. Zaidouni

TP/TD 1 : Gestion des processus – fork, exec wait, pipe

Les processus Unix

One to create them all !

Ouvrir un terminal et exécuter la commande `pstree`

1. A quoi sert le processus init ?

Solution:
C'est le père de tous les processus. A l'initialisation, il charge tous les processus systèmes soit l'horloge, le réseau, les drivers, montage des périphériques, etc.

2. A quel moment est-il exécuté ?

Solution:
Après le chargement du kernel. Dans l'ordre : BIOS -> BootLoader -> Kernel -> init

Création de processus

Modern families !

1. Que réalisent les programmes suivants ?
   Combien de processus sont créés ? Indiquez le résultat à l'écran, et dessinez le graphe des processus.

   1. ```
      int main(int argc, char **argv) {
              int p1, p2, p3;
              p1 = fork();
              p2 = getpid();
              p3 = getppid();
              printf("p1=%d - p2=%d - p3=%d\n",p1,p2,p3);
              return 0;
      }
      ```

   2. ```
      #define N 10
      int main(int argc, char **argv) {
              int i = 1;
              while (fork() == 0 && i <= N) i++;
              printf("%d\n",i);
              exit(0);
      }
      ```

   3. ```
      int main(int argc, char **argv) {
              int pid[3], i;
              for (i = 0; i < 3; ++i) {
                      pid[i] = fork();
              }
              printf("%d %d %d\n", pid[0], pid[1], pid[2]);
              return 0;
      }
      ```
      ```

Utilisation fork / exec

Gatcha !

1. écrire une commande `find_parallele` telle que `find_parallele` *fic rep* effectue la recherche d'un fichier de nom *fic* dans le répertoire *rep*. Pour cela, chaque processus parcourt le contenu du répertoire dont la référence lui est fournie ; s'il trouve un fichier de nom *fic*, il en affiche la référence complète ; et pour chaque sous-répertoire, il crée un processus fils chargé d'y poursuivre la recherche.

Solution:
Voir fichier `find_parallel.c`

Communication inter-processus (IPC) : les pipes

1. Que réalise le programme suivant ? Faire le schéma. Pourquoi mettre le chiffre 8 plutôt que 7 (ou 6) dans le write() ? Comment remplacer ce chiffre pour qu'on puisse changer facilement la taille du message à envoyer ?

```
int main(int argc, char **argv) {
        int p[2], pid;
        char buffer[16];

        if (pipe(p)<0) exit(1);

        pid = fork();
        if (pid<0) exit(1);
        if (pid==0) {
                close(p[0]);
                strcpy(buffer,"coucou\n");
                write(p[1],buffer,8);
        } else {
                close(p[1]);
                read(p[0],buffer,16);
                printf("J'ai lu: %s\n",buffer);
                waitpid(pid,0,0);
        }
}
```

2. Que se passerait-il si on mettait le fork() avant le pipe() ?
3. Écrire un programme qui crée un processus (en plus de lui-même). Le processus père envoie des chaînes de caractères à son fils, qui lui renvoie la taille de la chaîne. Le père affiche le résultat.

(Bonus) Redirection : dup2

1. Que fait la fonction `dup2` ?
2. Écrire un programme `countfiles` réalisant la commande `ls -l` *path* `| wc -l >` *file*. Pour cela, le processus père créera un fils exécutant `ls` et un autre exécutant `wc`.