TP à rendre 3

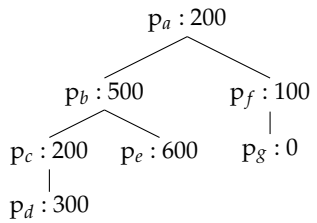
Rédigez un programme `genfork` qui prend en entrée une spécification de processus à générer. Votre programme admettra cette spécification soit sous forme d'un nom de fichier en argument, soit via l'entrée standard si aucun argument n'est fourni.

Ce fichier spécifie un arbre de processus à générer. Chaque ligne représente un sommet (un processus à générer) avec une valeur (une durée d'attente en millisecondes). Le nombre d'espaces (indentation) avant la valeur indique les arêtes, c'est-à-dire les relations de filiation des processus :

— le processus racine de l'arbre n'est précédé d'aucun espace ;
— les fils d'un processus indenté avec n espaces correspondent aux lignes suivantes indentées avec $n+1$ espaces, tant qu'elles ne sont pas séparées par une ligne indentée avec m espaces ($m \leq n$) ;
— si une ligne est indentée avec n espaces, la ligne immédiatement suivante est nécessairement indentée avec m espaces ($m \leq n+1$).

Par exemple, le fichier `specifork` ci-contre spécifie l'arbre représenté à droite : la racine de l'arbre est le processus p_a qui doit attendre 200 ms, il a pour fils directs les processus p_b et p_f ; le processus p_b a pour fils directs les processus p_c et p_e, et ainsi de suite.

```
200
␣500
␣␣200
␣␣␣␣300
␣␣600
␣100
␣␣0
```



Chacun des processus doit lancer ses fils directs, puis attendre le nombre de millisecondes spécifié (grâce à la fonction de bibliothèque `usleep`) et enfin attendre la terminaison de ses fils directs. Si tout se déroule sans erreur, la durée d'attente totale d'un processus correspond donc au maximum de la durée d'attente de tous les processus du sous-arbre.

On trouvera en annexe un exemple de l'utilisation de `genfork`.

On prendra comme convention que le code de retour de chaque processus (et de votre commande) correspond au nombre de processus générés si aucune erreur n'est détectée. Si une erreur est détectée (erreur système, spécification invalide, trop de processus générés, etc.), le code de retour vaudra 255. Si au moins un de ses fils signale une erreur, le père doit se terminer avec le code 255 pour répercuter l'erreur. On ne pourra bien sûr pas générer plus de 254 processus ; cependant, pour la lecture et l'analyse de la spécification, on ne fera aucune hypothèse sur le nombre maximum de processus.

Votre programme admettra éventuellement l'une des deux options (mutuellement exclusives) suivantes :

— `-d` (pour *debug*) : affichage de l'arbre, après lecture et analyse, sans générer de processus. L'arbre doit être affiché avec les mêmes conventions que pour la lecture ; autrement dit, l'arbre affiché doit être identique à l'arbre initial, à l'affichage des nombres près. Avec cette option, le code de retour devra être 0 en cas de succès, ou 255 en cas d'erreur.
— `-v` (pour *verbose*) : affichage d'une ligne pour chaque processus généré. Sans cette option, votre programme ne doit rien afficher si tout se passe bien.

Bien entendu, votre programme doit utiliser les primitives systèmes pour tout ce qui concerne la gestion des fichiers (notamment la lecture de la spécification) et des processus. Vous analyserez les options avec `getopt`. Votre programme doit compiler avec `cc -Wall -Wextra -Werror` (à l'exclusion de toute autre option) sur la machine turing. Les programmes qui ne compilent pas avec cette commande ne seront pas examinés.

Vous trouverez sur Moodle deux fichiers :
— Un fichier `genfork.c`, contenant des fonctions pour analyser une spécification complète fournie sous forme d'une chaîne de caractères unique et en extraire l'arbre correspondant, dont vous pouvez vous servir comme base pour votre propre programme ;
— Un script de test (`test-tp3.sh`).

Ce TP à rendre est individuel. On rappelle que la copie ou le plagiat sont sévèrement sanctionnés.

Annexe

Le résultat de l'exécution de l'exemple ci-dessus doit être comparable à :

```
turing$ ./genfork -d specifork
200
 500
  200
   300
  600
 100
  0

turing$ ./genfork -v specifork
pid 18738 -> 18739
pid 18738 -> 18740
pid 18738 : attente 200 ms
pid 18739 -> 18741
pid 18740 -> 18742
pid 18740 : attente 100 ms
pid 18742 : attente 0 ms
pid 18739 -> 18744
pid 18739 : attente 500 ms
pid 18741 -> 18743
pid 18742 <- 0 processus
pid 18741 : attente 200 ms
pid 18744 : attente 600 ms
pid 18743 : attente 300 ms
pid 18740 <- 1 processus
pid 18743 <- 0 processus
pid 18741 <- 1 processus
pid 18744 <- 0 processus
pid 18739 <- 3 processus
pid 18738 <- 6 processus

turing$ echo $?
7

turing$ time ./genfork -- < specifork
./genfork -- < specifork  0,00s user 0,00s system 1% cpu 0,606 total
```