Contrôle 1 : ASR31 - 2020

- [ ] Date : 1 octobre 2020
- [ ] Durée : 2h00
- [ ] Tous documents autorisés
- [ ] Les programmes seront dans un depot `git` de nom `ASR31Controles` (accessible par Denis Monnerat et Pierre Valarcher)
- [ ] les exercices sont dans des répertoires qui portent le nom des exercices en minuscules.

Exercice 1 (7pt) :

Ecrire des programmes `C` qui permettent de visualiser l'arborescence de processus suivante à la suite de l'appel `ps --forest` :

```
bash
 \_ A
     \_  B1 1
           \_ C1
    \_ B1 2
           \_ C2
```

ou bien

```
bash
 \_ A
     \_  B1 2
           \_ C2
    \_ B1 1
           \_ C1
```

Vous fournirez un `Makefile` et une commande shell qui permettra de visualiser l'arborescence.

Exercice 2 (13pt) :

En `C` les opérateurs binaires logique `||` et `&&` (le $\vee$ et le $\wedge$) sont séquentiels. L'exécution de `c1 || c2` se fait comme suit : on évalue `c1` puis si la valeur est `0` alors on évalue `c2` (si la valeur est différente de `0`, on n'évalue pas `c2`.
On veut implémenter une version parallèle de cette évaluation; on évalue en même temps `c1` et `c2` et dès que l'on sait que un des 2 est différent de `0` alors on arrête l'évaluation et on execute une commande. Sinon on attend que

les 2 évaluations se terminent avec comme valeur 0 et on exécute une autre commande.

1. On vous demande de proposer un plan de travail pour réaliser cela; décomposition en sous problèmes, plan de test (comment on teste les sous problèmes). Vous devez presque pouvoir écrire un sujet de TP qui amènera des étudiants de DUT2 (en cours d'ASR31) à réaliser le *ou parallèle*.
2. Vous mettrez en oeuvre la réalisation de ce *ou parallèle* de telle sorte que l'on puisse l'utiliser de la sorte :

```
$ time -p popor 3 oui 10 non 20 non 16
oui
real 10,2
...
$ time -p popor 2 oui 3 oui 1
oui
real 1,01
...
$ time -p popor 5 non 12 non 5 non 6 non 1 non 9
non
real 12,3
...
$
```

- `time` est la commande du shell qui calcule le temps mis par le programme passé en paramètre pour s'exécuter (le temps réel, utilisateur et appels système).
- `popor` est le nom du *poor parallel or*
- le premier argument est le nombre de processus à lancer en parallèle
- `non` *n* désigne un processus renvoyant la réponse faux (0) au bout de *n* secondes
- `oui` *n* désigne un processus renvoyant la réponse vraie (1) au bout de *n* secondes

Vous nous donnerez tous les outils (`Makefile`, script shell de test, readme.txt ou readme.md, planDeTravail.txt ou planDeTravail.md) pour que nous puissions évaluer votre travail avec bienveillance.