## Processus, signaux

**1°)** Soit l'extrait de programme suivant où les déclarations de variables n'apparaissent pas :

```
i=0 ;
printf(" Je suis le père processus : %d\n ",getpid()) ;
pid=fork() ;
i++ ;
printf(" Après 1er appel : %d et i =%d\n ",getpid(),i) ;
pid=fork() ;
i++ ;
printf(" Après 2ème appel : %d\et i =%d\n ",getpid(),i) ;
```

En supposant que :

Le processus père possède un pid égal à 1000
Seul notre processus fait des appels à fork() pendant tout le temps où l'on observe le système
Les numéros de processus sont distribués séquentiellement

a) Ne connaissant pas la politique d'ordonnancement, expliquer et indiquer, sans présumer de leur ordre, les affichages que nous pourrions obtenir. Une arborescence de l'exécution nous permettra d'expliquer le comportement de ce bout de programme.

**2°)** À la place du bit '**x**' associé au droit d'exécution lié à l'utilisateur propriétaire d'un fichier peut apparaître un bit '**s**'.

1°) Qu'elle en est la signification ?.
2°) Donner un exemple et commentez le.

**3°)** L'appel à la primitive wait() permet à un parent d'attendre la fin de l'un de ses enfants. L'appel à wait() peut se faire en asynchronisme total c'est à dire :

- avant la fin d'un quelconque enfant, et, dans ce cas là, l'appel est bloquant,
- après la fin d'au moins un enfant et on récupère alors le premier enfant à s'être signalé.

Lorsqu'il se termine, un enfant envoie le signal SIGCHLD à son parent. Sachant ceci, **que faut il faire**, pour qu'en synchronisme avec la fin de ses enfants, un processus parent, qui à plein d'autres choses à faire, affiche le message « Fin d'un enfant » au fur et à mesure où ceux-ci se terminent.

On donnera également, hors contexte, les quelques lignes en langage C qui permettent cette mise en œuvre.