DS INF204 2006–2007. Corrigé partiel.

Exercice 1 (4pts) Qu. 3 : par exemple `printf/system/exit/readdir/execlp` vs. `str**/mem**`.

Exercice 2 (2pts) 1. Elle retourne une variable locale automatique, le tableau et ses éléments sont alloués sur la pile, cette zone aura été désallouée au retour de la fonction `alloue_et_initialise` dans la fonction appelante. Warning à la compilation avec `gcc -Wall`, résultats imprédictibles à l'exécution.
2. `malloc()` (faute de mieux...).

Exercice 3 (3pts) 2. Les blocs ne sont pas supprimés (1) si le fichier a d'autres liens (2) s'il est encore ouvert par au moins un processus.
3. Séparer le `fork()` de l'`exec()` permet par exemple de faire simplement des redirections entre les 2.

Exercice 4 (5pts)

1. Une racine qui finit par échouer dans ses `fork()`, qui est donc dans l'état actif ou prêt, et plein de fils (moins que `sysconf(_SC_CHILD_MAX)`) passant zombis. Tuer le père termine le tout.
2. Une ligne. Chaque processus nouvellement créé fait un fils et passe en sommeil, en attente qu'il se termine. Le dernier processus échoue son `fork()`, mais reste dans la boucle (car son `fork()` retourne -1)). Il est le seul prêt/actif, les autres sont en sommeil sur `wait()`. Il suffit de terminer ce dernier processus, ce qui provoque le réveil en cascade des `wait()` et la terminaison de l'ensemble.
3. Arbre de forme quelconque a priori. Pour supprimer les processus, on les stoppe (`kill -STOP <pid>`), puis on les tue (`kill -INT <pid>` par exemple). Ceci suppose qu'on connaît chaque pid, et qu'on dispose d'un shell (et explique pourquoi les shells ont souvent une commande *interne* `kill`). Si on essaie de les tuer directement sans les stopper, un processus nouveau pourra être créé par l'un des noeuds de l'arbre (qui devient une forêt), et on n'a rien gagné.
4. Si aucun processus ne se termine avant la fin des créations, on obtient un arbre (représentant un tas binomial). On voit par récurrence que l'arbre B_N créé pour la valeur N est l'arbre

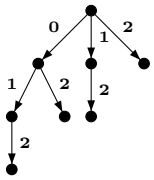

FIG. 1 – Arbre des processus

B_{N-1} auquel on a ajouté un fils gauche additionel, dont la racine est aussi B_{N-1}. Il y a donc $2^N = 8$ processus créés. La figure indique les parentés (flèches d'un père vers un fils), et sur les arêtes la valeur affichée avant le `fork()` correspondant. Il y a 4 affichages `2`, 2 affichages `1`, un affichage `0`. Par ailleurs, les processus se terminent d'eux-mêmes, un père pouvant se terminer avant ses fils car il n'y a pas de synchronisation.