TP1: Unix, programmation en C

Unix : prise en main

a. Ouvrir une session, se créer un compte, se reconnecter sur ce compte. Créer un répertoire `tds_systeme` puis un sous-répertoire `td1/`, tous les fichiers du tds seront sauvés dedans.

Utilisation du shell :

b. Se familiariser avec les commandes `ls`, `cd`, `mkdir`, `rm`, `echo`... (exemple : dans votre "home", créer un répertoire `foo`, y copier le fichier `/etc/passwd`, le regarder avec `less` puis l'effacer).

c. Qu'est-ce que la complétion automatique ? Essayer.

d. Que fait la commande `history` ? À l'aide de >, mettre le résultat de la commande dans un fichier. Utiliser `grep` pour voir quelles sont les commandes tapées contentant "ls". Faire la même chose avec |.

e. À l'aide de `du` et de `sort`, obtenez la liste des éléments de `/etc/` triés par taille. Que fait l'option `-n` de sort ?

Un premier programme en C

On rappelle qu'il faut sauver régulièrement les fichiers que l'on manipule.

a. Lancer un éditeur de texte (si vous ne savez pas quoi utiliser, vous pouvez essayer `emacs`). Pourquoi préfère-t-on taper `emacs &` plutôt que `emacs` ?

b. Écrire un programme simple qui contient une fonction `void main()` et qui affiche `bonjour`, le compiler et l'exécuter. Que font les options `-W` et `-Wall` de `gcc` et pourquoi faut-il toujours les utiliser ?

c. Créer un Makefile contenant une commande pour compiler et une commande pour effacer l'exécutable.

Printf, scanf, fonctions récursives

a. Regarder le man de la fonction `printf`.

b. Si `a` et `b` sont deux entiers, que vaut `a/b` ? Faire un programme qui affiche le résulat. Refaire la même chose avec deux a et b doubles.

c. Écrire une fonction récursive `int fact(int n)` et une `double fact_double(int n)` qui calcule $n!$ pour $n \geq 0$. Comparer `fact(20)` et `fact_double(20)` ? Comment expliquer ce résultat ?

d. À l'aide de scanf, faites un programme qui affiche "n=" puis vous affiche $n!$.

Un exemple de suite

Pour $a > 0$, on définit la suite $u_{n+1} = u_n/2 + a/(2u_n)$ et $u_0 = 1$.

a. Écrire une fonction `double rac(double a, int n)` qui calcule u_n. On fera attention aux cas où elle n'est pas définit ($n < 0$).

b. La tester pour $a = 1, 4, 9, \ldots$ et $n = 2, \ldots, 10$ Vers quoi converge-t-elle ?

Boucles

On pose $f_{n+2} = f_{n+1} + f_n$ et $f_0 = f_1 = 1$.

a. Écrire trois fonctions calculant f_n, l'une récursive, l'une utilisant une boucle `for` et l'une utilisant une boucle `while`. Quelles sont les différences ? Comparer la complexité, quelle est la solution à préférer ?

Arguments d'un programme

a. Faire un nouveau programme avec une fonction `int main(int argc, char ** argv)`. Que sont `argc` et `argv` ?

b. Faire un programme `add` qui additionne tous ses arguments et affiche le résultat (ex : `add 1 2.3 -1` devra afficher 2.3).