Analyse de la fonction `qsort` de la `glibc`

$\pi\lambda$

Octobre 2013
dernière compilation : 7 novembre 2014

Résumé

Dans le cours, nous avons établi que le tri rapide `qsort` traite, en moyenne, un tableau de taille n en utilisant $2n \log n$ comparaisons. Un fait facile à mettre en évidence au travers d'une expérience numérique introductive à : `qsort`, `/usr/bin/time`, `gnuplot`, `make`, `gcc`, `bash`.

1 mesure des temps

On dispose de plusieurs solutions pour faire des mesures de temps de calcul :
— à la source : `time()`, `clock()`, `gettimeofday()` ...
— la commande interne `time`.
— le profileur `gprof`.
— la commande externe `/user/bin/time`.

Avec la dernière option, on peut lancer le chronométrage d'une commande, en précisant un format d'affichage :

```
sample> /usr/bin/time --format="%E" sleep 7
0:07.02
sample> /usr/bin/time --format="%E" sleep 1 | tr : M
0:01.00
sample> /usr/bin/time --format="%E" sleep 1 |& tr : M
0M01.00
```

On notera la différence avec la commande interne `time` qui mesure le temps de caclcul d'un pipeline :

```
sample> time sleep 1 | grep user

real 0m1.008s
user 0m0.000s
sys 0m0.004s
sample> time sleep 1 |&  grep user

real 0m1.005s
user 0m0.000s
```

TABLE 1 – Les principales options du format d'affichage de la commande `/usr/bin/time`

FMT	détails
%E	Elapsed real time (in [hours :]minutes :seconds)
%e	(Not in tcsh.) Elapsed real time (in seconds)
%S	Total number of CPU-seconds spent in kernel mode
%U	Total number of CPU-seconds spent in user mode
%P	Ratio CPU
%M	Memory size
%F	Number of major page faults.
%R	Number of minor, or recoverable, page faults.
%W	Number of times the process was swapped out of main memory
%c	Number of times the process was context-switched

```
TIME(1) Linux User's Manual
NAME
  time - time a simple command or give resource usage
SYNOPSIS
       time [options] command [arguments...]
DESCRIPTION
       The time command runs the specified program command
with the given arguments. When command finishes, time writes
a message to standard error giving timing statistics about
this program run.
       Note: some shells (e.g., bash(1)) have a built-in time
command that provides less functionality than the
    command
described here. To access the real command, you may need to
specify its pathname (something like /usr/bin/time).
```

```
sys 0m0.003s
sample> (time sleep 1) |&  grep user
user 0m0.000s
```

On constate que, par défaut, le résultat de la mesure est envoyé sur la sortie erreur standard. Les principales options du format de la commande `/usr/bin/time` sont résumées dans la table TAB. [**??**].

2 makefile

On implante le tri rapide de la bibiothèque standard, dans une source `qsort.c`. Le document que vous lisez est contruit à partir de cette source, les compilations sont pilotées par un fichier `makefile`. Les principaux ingrédients d'une expérience numérique y sont présents :

— le `makefile` et ses cibles.

TABLE 2 – Les principales options de la commande `make`.

option	
`make`	pour satifaire la première cible
`make -d`	pour voir les tentatives implicites
`make -B`	pour forcer les compilations
`make cible`	pour satisfaire cible
`make -C`	changer de répértoire
`make X=13`	initialisée la variable X

— la compilation des objets.
— la compilation d'un exécutable.
— un script pour réaliser les mesures.
— `gnuplot` pour dessiner.
— `pdflatex` pour compiler la source LaTeX.

Rappelons que `make`, ouvre un fichier `makefile` du répertoire courant pour mettre à jour la première cible en fonction de certaines action sous des conditions de dépendances.

```
cible : dependances
[tabulation] action 1
[tabulation] action 2
```

Les utilisations courantes de la commande `make` sont résumées dans TAB. [??].

Dans l'exemple,
— `make nice` met les sources C dans le style de codage dit Kernighan & Richie.
— `make clean` efface les traits de constructions.
— `make proper` pour repartir sur de nouvelles bases.

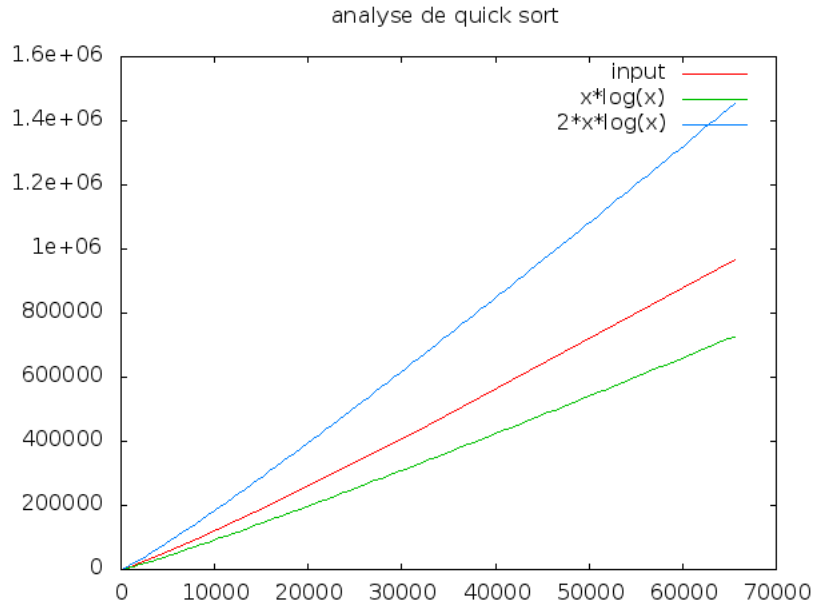
```
SHELL=/bin/bash
CFLAGS=-Wall -g
all      : count.pdf
count.pdf : count.png count.tex
 pdflatex count.tex

qsort.exe : table.o qsort.c
 gcc $(CFLAGS) table.o qsort.c -o qsort.exe

table.o : table.c
 gcc $(CFLAGS) -c table.c

count.txt : qsort.exe
 for l in {1..16}; do\
          ./qsort.exe -l $$l -c;\
        done >  count.txt
```

```
count.png : count.plt count.txt
 sed -n 's/[a-z]*=/ /gp' count.txt > count.dat
 gnuplot count.plt

clean :
 rm -f *.log *.aux
 rm -f *~

proper:
 rm -f *.exe *.dat *.png *.o
 make clean
nice  :
 indent -kr *.c
tar   :
 make proper
 tar cvf ../sample.tar *
```

3 graphique

Le graphique est construit par `gnuplot` à partir d'un fichier de commandes `count.plt`, et d'un fichier de données `count.dat`.

```
set output 'count.png'
input = "count.dat"
set terminal png
set title 'analyse de quick sort'
plot input with lines, x*log(x), 2*x*log(x)
quit
```



1. Reproduire le graphique analogue.
2. Comparer ces résultats avec ceux du cours.

4 qsort

Le programme proposé utilise deux fonctions importantes des bibliotèques standards : `getopt` pour gérer les options de la ligne de commande et la fonction de tri générique `qsort`.

```
1  #include "table.h"
2
3
4  int comp = 0;
5
6  int cmp(const void *x, const void *y)
7  {
8      comp++;
9      if (*(int *) x < *(int *) y)
10         return -1;
11     if (*(int *) x > *(int *) y)
12         return +1;
13     return 0;
14 }
15
```

```c
int main(int argc, char *argv[])
{
    int i, n = 0, *t;
    int opt, count = 0, skip = 0;
    while ((opt = getopt(argc, argv, "l:n:cs")) != -1) {
        switch (opt) {
        case 'l':
            n = 1 << atoi(optarg);
            break;
        case 'n':
            n = atoi(optarg);
            break;
        case 'c':
            count = 1;
            break;
        case 's':
            skip = 1;
            break;
        default:                    /* '?' */
            fprintf (stderr, "Usage: %s -ln{number} -count -skip\n",
                    argv[0]);
            exit (1);
        }
    }
    srandom( time(NULL));
    t = calloc(n, sizeof(t));
    for (i = 0; i < n; i++)
        t[i] = random();
    if ( ! skip) {
        qsort(t, n, sizeof(int), cmp);
        if (count)
            printf ("\nn=%d count=%d", n, comp);
    }
    return 0;
}
```

1. Quelles sont les options gérées dans `qsort.c` ?
2. Reproduire l'exemple `man 3 qsort`.
3. Modifier cet exemple pour trier des nombres de petites tailles passés par la ligne de commande.

5 code source

Obtenir le code source de la fonction `qsort` de la `glibc`.

1. Comment est implantée la récursion ?

2. Comment est implantée la coupure ?
3. Quelle est la valeur de seuil pour la récursion ?

6 `sort`

1. Utiliser le `ltrace` pour vérifier que `sort` utilise `qsort`.
2. Mesurer le temps de calcul de la commande externe `sort` sur des fichiers aléatoires.

— Ecrire une commande pour trier les lignes d'un fichier.
— Comparer les performances avec `sort`.
— Comparer avec un script `python`.
— Comparer avec un script `perl`.