

Lab 1 report

6314061 – Quang Tuan Le

January 2023

1 Task 1: Learning how to write a Makefile

Contents of makefile:

```
APP=lab1
all:
    gcc -pg ${APP}.c -o ${APP}
run:all
    ./${APP}
clean:
    rm ${APP}
```

2 Task 2: Using gettimeofday to measure performance

Following the Lab 1 instruction video, I ran the code with 3 different input values (1000, 10000, and 100000), and I used gettimeofday() function to get the elapsed time. So the output time below is the average time of 10 times which I ran the myLoop function. I also attached the code file to the submission.

1. Input: 1000, Output: 0.000003 SECONDS
2. Input: 10000, Output: 0.000029 SECONDS
3. Input: 100000, Output: 0.000253 SECONDS

3 Task 3: Performance profiling with gprof

For this task, I followed the Lab 1 instruction video, and I changed the input value to 100,000,000. Then, I got the below result.

3.1 Task 3.1:

The topmost time consuming routines are: (Flat profile)

- myLoop: 84.77% of time ~ 2.08 seconds

output:

```
quangle — ssh qle010@onyx.cs.fiu.edu — 85x33
Flat profile:
█
Each sample counts as 0.01 seconds.
%   cumulative   self           self       total
time  seconds    seconds   calls   ms/call  ms/call  name
84.77     2.08      2.08        10    207.68   207.68  myLoop

%           the percentage of the total running time of the
time        program used by this function.

cumulative  a running sum of the number of seconds accounted
seconds    for by this function and those listed above it.

self        the number of seconds accounted for by this
seconds     function alone.  This is the major sort for this
            listing.

calls       the number of times this function was invoked, if
            this function is profiled, else blank.

self        the average number of milliseconds spent in this
ms/call     function per call, if this function is profiled,
            else blank.

total       the average number of milliseconds spent in this
ms/call     function and its descendents per call, if this
            function is profiled, else blank.

name        the name of the function.  This is the minor sort
            for this listing.  The index shows the location of
            the function in the gprof listing.  If the index is
            in parenthesis it shows where it would appear in
```

3.2 Task 3.2:

Output: (Call graph)



Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) for 0.48% of 2.08 seconds

index	% time	self	children	called	name
		2.08	0.00	10/10	main [2]
[1]	100.0	2.08	0.00	10	myLoop [1]

					<spontaneous>
[2]	100.0	0.00	2.08		main [2]
		2.08	0.00	10/10	myLoop [1]

This table describes the call tree of the program, and was sorted by the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the index number at the left hand margin lists the current function. The lines above it list the functions that called this function, and the lines below it list the functions this one called.

This line lists:

index	A unique number given to each element of the table. Index numbers are sorted numerically. The index number is printed next to every function name so it is easier to look up where the function is in the table.
% time	This is the percentage of the 'total' time that was spent in this function and its children. Note that due to different viewpoints, functions excluded by options, etc, these numbers will NOT add up to 100%.

- **myLoop** spawns no children, 2.08 seconds spent here.
- **main** spawns 1 children (myLoop), 2.08 seconds spent here.