

# Contents

1	Basic Test Results	2
2	README	3
3	Graph.png	4
4	Makefile	5
5	osm.cpp	6

# 1 Basic Test Results

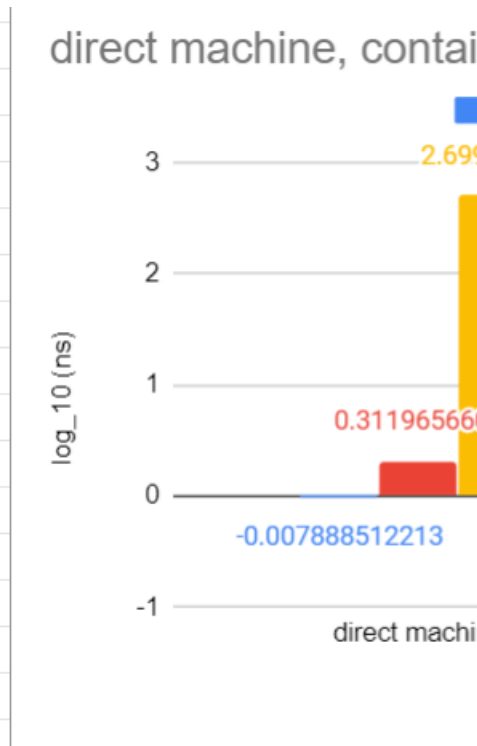
```
1 ['brahan', 'danielabayev']
2 make test
3 passed basic make test
4 compile test
5 passed basic compile test
```

## 2 README

```
1  brahan, danielabayev
2  Brahan Wassan(320455116), Daniel Abayev (206224396)
3  EX: 1
4
5  FILES:
6  osm.c -- a static library with the programming part of the ex
7  makefile -- a makefile for the program
8  os_ex1_part_b.png
9
10 REMARKS:
11 No remarks
12
13 ANSWERS:
14
15 Assignment 1 - Using strace to understand what a program is doing:
16
17 If you run the command without execute permission you get :
18     strace: exec: Permission denied\n", 32strace: exec: Permission denied
19 And the program exits
20 After granting the program execute permission you get:
21
22 If you running the program any number of arguments that different from one you get:
23     "Error. The program should receive a single argument. Exiting.\n: Success\n"
24 Meaning the program writing an error message and then exits
25
26 If you run the program with exec one argument you get:
27     mkdir("Welcome", 0775)                = 0
28     mkdir("Welcome/To", 0775)              = 0
29     openat(AT_FDCWD, "Welcome/To/OS2020", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
30     fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
31     write(3, "brahan\nIf you haven't read the course guidelines yet --- do it right now!\nempty", 79) = 79
32     close(3)                              = 0
33     unlink("Welcome/To/OS2020")            = 0
34     rmdir("Welcome/To/")                   = 0
35     rmdir("Welcome/")                      = 0
36     exit_group(0)
37
38 Meaning the program creates a directory named "Welcome" with mkdir
39 Inside of it it creates another directory named To
40 Then its create a file called OS2020 (if its already created then it's just open it)
41 Inside the new file it writes:
42     "USER_NAME\nIf you haven't read the course guidelines yet --- do it right now!\nGIVEN_INPUTFILE"
43 Then we delete the OS2020 file with unlink and the directory with rmdir and exit the program.
44
45
```

### 3 Graph.png

	log scale			
	direct machine	container	vm	
op	-0.007888512213	0.00346053211	0.7891574919	
empty_function	0.3119656604	0.3121773564	1.444138518	
sys_call	2.699594071	2.706518934	2.918626296	
	direct machine	container	vm	
op	0.982	1.008	6.154	
empty_function	2.051	2.052	27.806	
sys_call	500.719	508.767	829.137	



## 4 Makefile

```
1  CC=g++
2  CXX=g++
3  RANLIB=ranlib
4
5  LIBSRC=osm.cpp
6  LIBOBJ=$(LIBSRC:.cpp=.o)
7
8  INCS=-I.
9  CFLAGS = -Wall -std=c++11 -g $(INCS)
10 CXXFLAGS = -Wall -std=c++11 -g $(INCS)
11
12 OSMLIB = libosm.a
13 TARGETS = $(OSMLIB)
14
15 TAR=tar
16 TARFLAGS=-cvf
17 TARNAME=os_ex1.tar
18 TARSRC=$(LIBSRC) Makefile README
19
20 all: $(TARGETS)
21
22 $(TARGETS): $(LIBOBJ)
23     $(AR) $(ARFLAGS) $@ $^
24     $(RANLIB) $@
25
26 clean:
27     $(RM) $(TARGETS) $(OSMLIB) $(OBJ) $(LIBOBJ) *~ *core
28
29 depend:
30     makedepend -- $(CFLAGS) -- $(SRC) $(LIBSRC)
31
32 tar:
33     $(TAR) $(TARFLAGS) $(TARNAME) $(TARSRC) Graph.png
```

## 5 osm.cpp

```
1  #include <sys/time.h>
2  #include "osm.h"
3
4  #define ERR -1
5  #define NOT_VALID 0
6  #define UNROLL_FACTOR 5
7  struct timeval start, end;
8
9  unsigned int loop_unrolling_factor(const unsigned int &iterations) {
10     unsigned int real_iterations = iterations;
11     unsigned int reminder = iterations % UNROLL_FACTOR;
12     if (reminder != 0) {
13         real_iterations += reminder; // round factor
14     }
15     return real_iterations;
16 }
17
18 double calculate_time_taken() {
19     double time_taken = (end.tv_sec - start.tv_sec) * 1e6;
20     return (double) (time_taken + (end.tv_usec - start.tv_usec)) * 1e3;
21 }
22
23 double osm_operation_time(unsigned int iterations) {
24     if (iterations == NOT_VALID) {
25         return ERR;
26     } else {
27         int a = 0, b = 0, c = 0;
28         unsigned int real_iterations = loop_unrolling_factor(iterations);
29         int is_valid = gettimeofday(&start, nullptr);
30         if (is_valid != ERR) {
31             for (unsigned int i = 0; i < real_iterations; i = i + UNROLL_FACTOR) {
32                 a = a + 1; //loop unrolling , factor 5
33                 b = b + 1;
34                 c = c + 1;
35                 a = a + 1;
36                 b = b + 1;
37             }
38             is_valid = gettimeofday(&end, nullptr);
39             if (is_valid != ERR) {
40                 return calculate_time_taken() / (double) (real_iterations);
41             }
42         }
43
44         return ERR;
45     }
46 }
47
48 void empty_function() {}
49
50 double osm_function_time(unsigned int iterations) {
51     if (iterations == NOT_VALID) {
52         return ERR;
53     } else {
54         unsigned int real_iterations = loop_unrolling_factor(iterations);
55         int is_valid = gettimeofday(&start, nullptr);
56         if (is_valid != ERR) {
57             for (unsigned int i = 0; i < real_iterations; i = i + UNROLL_FACTOR) {
58                 empty_function();
59                 empty_function();
60             }
61         }
62         return ERR;
63     }
64 }
```

```

60         empty_function();
61         empty_function();
62         empty_function();
63     }
64     is_valid = gettimeofday(&end, nullptr);
65     if (is_valid != ERR) {
66         return calculate_time_taken() / (double) (real_iterations);
67     }
68 }
69
70 return ERR;
71 }
72 }
73
74 double osm_syscall_time(unsigned int iterations) {
75     if (iterations == NOT_VALID) {
76         return ERR;
77     } else {
78         unsigned int real_iterations = loop_unrolling_factor(iterations);
79         int is_valid = gettimeofday(&start, nullptr);
80         if (is_valid != ERR) {
81             for (unsigned int i = 0; i < real_iterations; i = i + UNROLL_FACTOR) {
82                 OSM_NULLSYSCALL;
83                 OSM_NULLSYSCALL;
84                 OSM_NULLSYSCALL;
85                 OSM_NULLSYSCALL;
86                 OSM_NULLSYSCALL;
87             }
88             is_valid = gettimeofday(&end, nullptr);
89             if (is_valid != ERR) {
90                 return calculate_time_taken() / (double) (real_iterations);
91             }
92         }
93         return ERR;
94     }
95 }

```