Contents

1 Basic Test Results 2

2 README 3

3 Makefile 4

4 ex1.png 5

5 osm.h 6

6 osm.cpp 7

1 Basic Test Results

```
['davidponar']
make test
passed basic make test
compile test
passed basic compile test
```

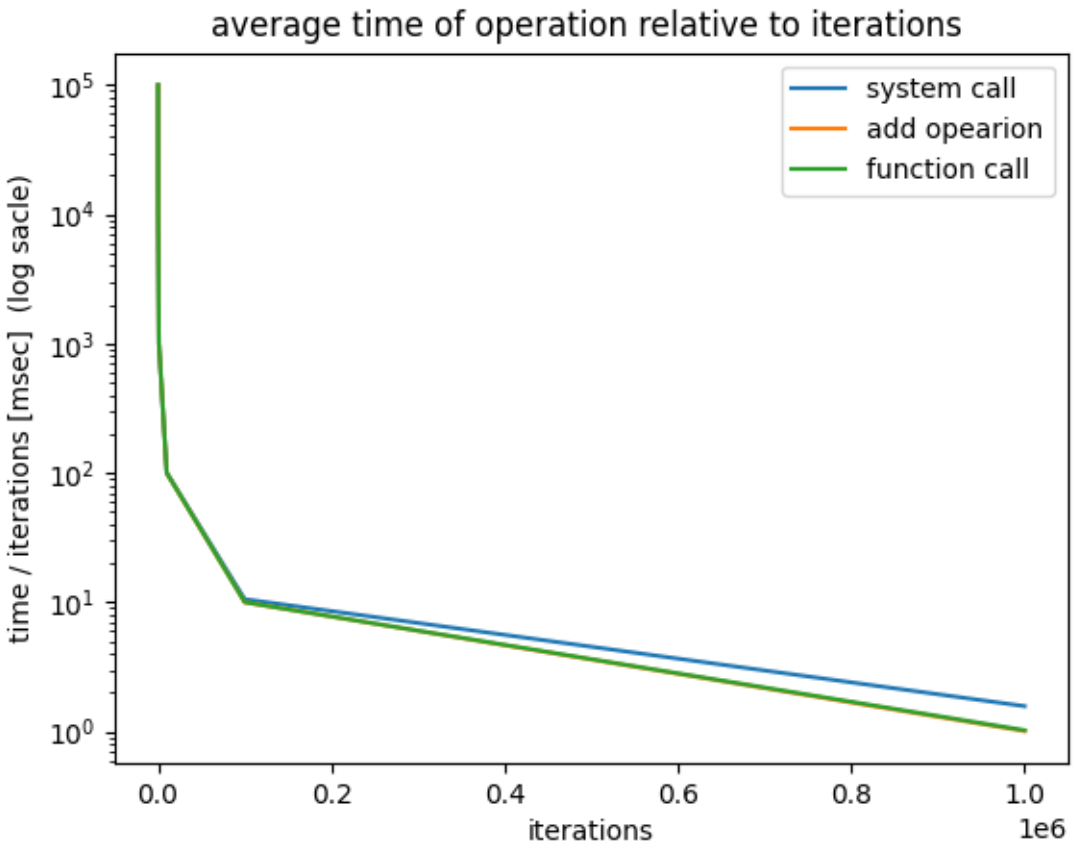
2 README

```
davidponar
David Ponarovsky 208504050
EX: 1

FILES:
osm.cpp -- a file with some code
osm.h -- header file
Makefile -- the make file
ex1.png -- the png file.

ANSWERS:
Q1:

(1) first the program ask for an argument, if there isn't one then it's kill it's self.
(2) then it uses system calls mkdir() tom create a path.
(3) inside the directory the program use the opanat() system call to create a file
(4) then it's uses the fstat() system call to obtain a meta data, from there it's extract the uid or user_name
(5) then the program use's that name for output's some string which include's my name
(6) finaly removes the file, directory and unlinks the memmory via rmdir , unlink systems call
```

3 Makefile

```makefile
CC=g++
CXX=g++
RANLIB=ranlib

LIBSRC=osm.cpp
LIBOBJ=$(LIBSRC:.cpp=.o)

INCS=-I.
CFLAGS = -Wall -std=c++11 -g $(INCS)
CXXFLAGS = -Wall -std=c++11 -g $(INCS)

OSMLIB = libosm.a
TARGETS = $(OSMLIB)

TAR=tar
TARFLAGS=-cvf
TARNAME=ex1.tar
TARSRCS=$(LIBSRC) Makefile README ex1.png osm.h

all: $(TARGETS)

$(TARGETS): $(LIBOBJ)
	$(AR) $(ARFLAGS) $@ $^
	$(RANLIB) $@

clean:
	$(RM) $(TARGETS) $(OSMLIB) $(OBJ) $(LIBOBJ) *~ *core

depend:
	makedepend -- $(CFLAGS) -- $(SRC) $(LIBSRC)

tar:
	$(TAR) $(TARFLAGS) $(TARNAME) $(TARSRCS)
```

4 ex1.png


average time of operation relative to iterations
system call
add opearion
function call
time / iterations [msec] (log sacle)
10^5
10^4
10^3
10^2
10^1
10^0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
iterations

5 osm.h

```c
#ifndef _OSM_H
#define _OSM_H


/* calling a system call that does nothing */
#define OSM_NULLSYSCALL asm volatile( "int $0x80 " : : \
        "a" (0xffffffff) /* no such syscall */, "b" (0), "c" (0), "d" (0) /*:\
        "eax", "ebx", "ecx", "edx"*/)


/* Time measurement function for a simple arithmetic operation.
   returns time in nano-seconds upon success,
   and -1 upon failure.
   */
double osm_operation_time(unsigned int iterations);


/* Time measurement function for an empty function call.
   returns time in nano-seconds upon success,
   and -1 upon failure.
   */
double osm_function_time(unsigned int iterations);


/* Time measurement function for an empty trap into the operating system.
   returns time in nano-seconds upon success,
   and -1 upon failure.
   */
double osm_syscall_time(unsigned int iterations);


#endif
```

6 osm.cpp

```cpp
#ifndef _OSM_H
#include "osm.h"
#define _OSM_H
#endif
#include "sys/time.h"

/* calling a system call that does nothing */
#define OSM_NULLSYSCALL asm volatile( "int $0x80 " : : \
        "a" (0xffffffff) /* no such syscall */, "b" (0), "c" (0), "d" (0) /*:\
        "eax", "ebx", "ecx", "edx"*/)



#define MEASURING( Code )\
    struct timeval t1, t2;\
    double elapsedTime;\
    gettimeofday(&t1, nullptr);\
    for (unsigned int i = 0; i < iterations; i++)\
    {                                                \
        Code                                         \
    }                                                \
    gettimeofday(&t2, nullptr);                         \
    elapsedTime = (t2.tv_sec - t1.tv_sec) * 1000.0; \
    return (elapsedTime + (t2.tv_usec - t1.tv_usec)) / iterations;



/* Time measurement function for a simple arithmetic operation.
   returns time in nano-seconds upon success,
   and -1 upon failure.
   */
double osm_operation_time(unsigned int iterations)
{
    // credit for stackoverflow.

    int a = 100;
    int b = 243;
    MEASURING( a + b; )
}

void empty_function_call()
{

}


/* Time measurement function for an empty function call.
   returns time in nano-seconds upon success,
   and -1 upon failure.
   */
double osm_function_time(unsigned int iterations)
{
    MEASURING(empty_function_call();)
}
/* Time measurement function for an empty trap into the operating system.
   returns time in nano-seconds upon success,
   and -1 upon failure.
   */
double osm_syscall_time(unsigned int iterations)
```

```
{
    MEASURING(OSM_NULLSYSCALL; )
}
```