



Name : Shubhashree Honnagangaiah Raghu Raja

CSU ID : 2859986

Grail ID : [shonnag@grail.eecs.csuohio.edu](mailto:shonnag@grail.eecs.csuohio.edu)

## Introduction:

Aim here is to create four c codes for etime.c, et.c ep.c and para\_mm.c. Where etime.c is for etime() function which will return elapsed time since the last call to it, et.c is to call pthread\_create() or pthread\_join() to evaluate time for thread creation or deletion, ep.c is to measure the process creation or deletion time by invoking fork() or waitpid() and para\_mm.c to measure computation time for multiplying two 400\*400 matrix.

I have to get elapsed time when I execute et.c, ep.c and para\_mm.c with etime.c and also when compiled and linked it should build respective executable file that is ep, et and para\_mm.

## Description for my codes

### etime.c

This code is implementing a function etime() that calculates the elapsed time since the last time etime() was called. Here's a breakdown of what's happening:

- `#include <sys/time.h>` and `#include <stdlib.h>`
  - These lines include the necessary header files for using functions related to time and memory allocation.
- `struct timeval start;`
  - This line declares a variable start of type struct timeval. This structure is commonly used to represent time with microsecond precision.
- `float etime();`
  - This line declares a function named etime() that returns a float. This function calculates the elapsed time since the last call to etime().
- Inside the etime() function:
  - `struct timeval end;`
    - Declares a variable end of type struct timeval. This will store the current time when etime() is called.
  - `float Seconds, Useconds, elapsed_time;`
    - Declares variables to store seconds, microseconds, and the total elapsed time.
  - `gettimeofday(&end, NULL);`
    - Retrieves the current time and stores it in the end variable.
  - `Seconds = (end.tv_sec - start.tv_sec);`
    - Calculates the number of seconds elapsed since the last call by subtracting the tv\_sec member of end from the tv\_sec member of start.
  - `Useconds = (end.tv_usec - start.tv_usec);`
    - Calculates the number of microseconds elapsed since the last call by subtracting the tv\_usec member of end from the tv\_usec member of start.
  - `elapsed_time = (Seconds + Useconds) / 1000000.0;`
    - Calculates the total elapsed time in seconds by adding the seconds and microseconds and converting the result to seconds.

- `start = end;`
  - Updates the start time to the current time stored in `end`.
- `return elapsed_time;`
  - Returns the total elapsed time in seconds.

In summary, this code allows you to calculate the elapsed time between calls to the `etime()` function with microsecond precision. The first call to `etime()` will return the time elapsed since the program started, and subsequent calls will return the time elapsed since the previous call.

### et.c

- Header Inclusions
  - `#include <stdio.h>`: Provides input/output functionalities like `printf()` and `scanf()`.
  - `#include <stdlib.h>`: Includes functions like `malloc()` and `free()` for memory allocation and deallocation.
  - `#include <string.h>`: Provides string manipulation functions like `strcmp()`.
  - `#include <pthread.h>`: Necessary for multi-threading functionalities.
  - `extern float etime();`: Declares an external function `etime()` which is declared in `etime.c`.
- Global Variables
  - `int size`: This variable will hold the size of the buffer (in kilobytes).
  - `char* buffer`: Pointer to a buffer where memory operations will be performed.
  - `char* before_or_after_fork`: Indicates whether memory operations happen before or after forking a process.
- Thread Function:
  - `void *thread_function(void *arg)`: This function is executed by a pthread. It initializes the buffer with a value of 100 if `before_or_after_fork` is set to `"-a"`.
- Main Function:
  - It starts by checking if the number of command-line arguments is not equal to 3. If incorrect, it prints the usage and exits.
  - Parses command-line arguments:
    - `argv[1]` is expected to be either `"-a"` or `"-b"`, indicating whether memory operations occur before or after forking.
    - `argv[2]` represents the size of the buffer in kilobytes.
  - Initializes variables:
    - Converts `argv[2]` to an integer and assigns it to `size`.
    - Allocates memory for the buffer using `calloc()` with `size * 1024` as the size.
    - Assigns `argv[1]` to `before_or_after_fork`.
  - Measures the start time using `etime()`.
  - Depending on the value of `before_or_after_fork`:
    - If it's `"-b"`, it initializes the buffer with a value of 100 in 4096-byte increments.
    - If it's `"-a"`, it creates a pthread to execute `thread_function()` in parallel.

# Project 1 Report

- Waits for the pthread to finish using pthread\_join().
- Measures the elapsed time using etime() and prints it out.

## Output of et.c and etime.c

The image displays a remote terminal window titled 'spirt.eecs.csuohio.edu'. The terminal shows a series of commands and their outputs. The first command is 'et -a 0', followed by a loop of 'et -a 1024' and 'et -a 16384'. The output for each command shows the total elapsed time in seconds. The terminal is running on a system with a root directory of '/home/student/shhonnag/proj1/'. The left sidebar shows a file explorer with a tree view containing 'ep', 'et', 'etmc.c', 'etmc.o', 'make', 'para\_rm', and 'para\_rm.c'. The bottom status bar indicates the system is 'Mostly sunny' and the time is 15:11 on 14-02-2024. The terminal output shows a sequence of commands and their outputs, including 'et -a 0', 'et -a 1024', 'et -a 16384', and a loop of 'et -a 1024' and 'et -a 16384'. The output for each command shows the total elapsed time in seconds.

```

shhonnag@spirit:~/proj1$ ./et -b 0
Total elapsed time : 0.000126 seconds
shhonnag@spirit:~/proj1$ ./et -b 1024
Total elapsed time : 0.000302 seconds
shhonnag@spirit:~/proj1$ ./et -b 8192
Total elapsed time : 0.000367 seconds
shhonnag@spirit:~/proj1$ ./et -b 16384
Total elapsed time : 0.000215 seconds
shhonnag@spirit:~/proj1$ ./et -b 32768
Total elapsed time : 0.000265 seconds

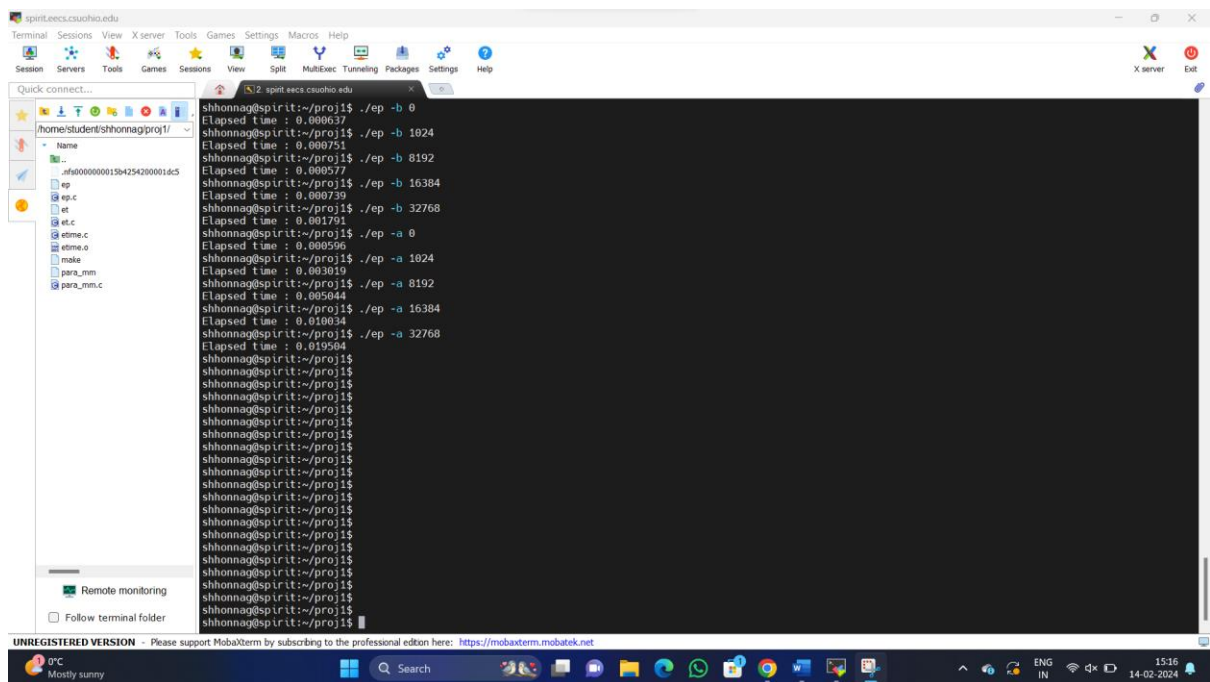
```

## ep.c

- Header Inclusions
  - `#include <stdio.h>`: Provides input/output functionalities like `printf()` and `scanf()`.
  - `#include <stdlib.h>`: Includes functions like `malloc()` and `free()` for memory allocation and deallocation.
  - `#include <unistd.h>`: Provides access to POSIX operating system API for system calls like `fork()`.
  - `#include <string.h>`: Provides string manipulation functions like `strcmp()`.
  - `#include <sys/types.h>`: Defines various data types used in system calls and library functions.
  - `#include <sys/wait.h>`: Provides functions for waiting on child processes.
- External Function Declaration
  - `extern float etime();`: Declares an external function `etime()` which is expected to measure elapsed time. This function is defined in `etime.c`.
- Main Function:
  - It starts by checking if the number of command-line arguments is not equal to 3. If incorrect, it prints the usage and exits.
  - Parses command-line arguments:
    - `argv[1]` is expected to be either `"-a"` or `"-b"`, indicating whether memory operations occur before or after forking.
    - `argv[2]` represents the size of the buffer in kilobytes.
  - Initializes variables:
    - Converts `argv[2]` to an integer and assigns it to `size`.
    - Allocates memory for the buffer using `calloc()` with `size * 1024` as the size.
  - Measures the start time using `etime()`.
  - Depending on the value of `argv[1]`:
    - If it's `"-b"`, it initializes the buffer with a value of 100 in 4096-byte increments.
    - If it's `"-a"`, it forks a child process and initializes the buffer in the child process.
  - In the parent process:
    - It waits for the child process to finish using `waitpid()`.
    - Measures the elapsed time using `etime()` after the child process completes.
    - Prints the elapsed time.

## Project 1 Report

### Output of ep.c and etime.c



```
shhonnag@spirit:~/proj1$ ./ep -b 0
Elapsed time : 0.000637
shhonnag@spirit:~/proj1$ ./ep -b 1024
Elapsed time : 0.000751
shhonnag@spirit:~/proj1$ ./ep -b 8192
Elapsed time : 0.000577
shhonnag@spirit:~/proj1$ ./ep -b 16384
Elapsed time : 0.000739
shhonnag@spirit:~/proj1$ ./ep -b 32768
Elapsed time : 0.001791
shhonnag@spirit:~/proj1$ ./ep -a 0
Elapsed time : 0.000596
shhonnag@spirit:~/proj1$ ./ep -a 1024
Elapsed time : 0.003019
shhonnag@spirit:~/proj1$ ./ep -a 8192
Elapsed time : 0.005044
shhonnag@spirit:~/proj1$ ./ep -a 16384
Elapsed time : 0.010034
shhonnag@spirit:~/proj1$ ./ep -a 32768
Elapsed time : 0.010504
```

```
shhonnag@spirit:~/proj1$ ./ep -b 0
Elapsed time : 0.000637
shhonnag@spirit:~/proj1$ ./ep -b 1024
Elapsed time : 0.000751
shhonnag@spirit:~/proj1$ ./ep -b 8192
Elapsed time : 0.000577
shhonnag@spirit:~/proj1$ ./ep -b 16384
Elapsed time : 0.000739
shhonnag@spirit:~/proj1$ ./ep -b 32768
Elapsed time : 0.001791
shhonnag@spirit:~/proj1$ ./ep -a 0
Elapsed time : 0.000596
shhonnag@spirit:~/proj1$ ./ep -a 1024
Elapsed time : 0.003019
shhonnag@spirit:~/proj1$ ./ep -a 8192
Elapsed time : 0.005044
shhonnag@spirit:~/proj1$ ./ep -a 16384
Elapsed time : 0.010034
shhonnag@spirit:~/proj1$ ./ep -a 32768
Elapsed time : 0.010504
```



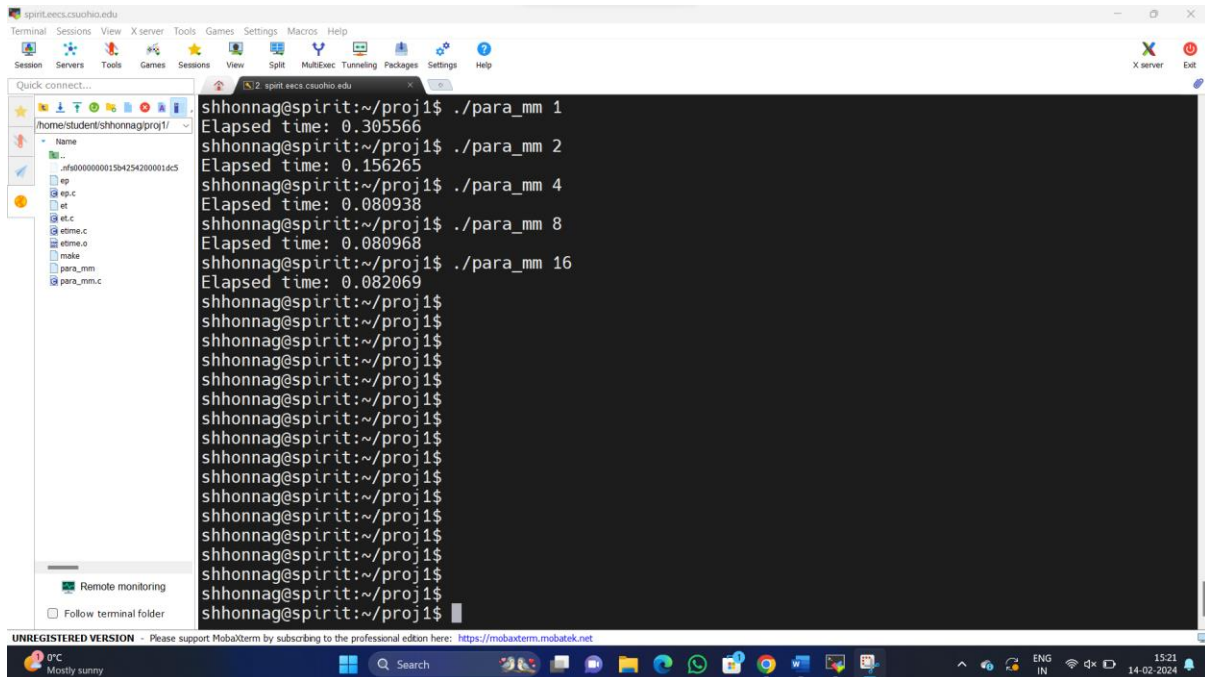
## para\_mm.c

- Header Inclusions:
  - `#include <stdio.h>`: Provides input/output functionalities like `printf()` and `scanf()`.
  - `#include <stdlib.h>`: Includes functions like `malloc()` and `free()` for memory allocation and deallocation.
  - `#include <sys/time.h>`: Defines structures and functions for handling time-related operations.
  - `#include <pthread.h>`: Provides functionalities for multi-threading.
  - `#include <string.h>`: Provides string manipulation functions like `strcmp()`.
  - `#include <semaphore.h>`: Defines semaphores for thread synchronization.
- External Function Declaration
  - `extern float etime();` Declares an external function `etime()` which is expected to measure elapsed time. This function is defined in `etime.c`
- Global Variables
  - `sem_t semaphore`: Semaphore used for synchronization.
  - `int num_threads`: Number of threads to be used for parallel execution.
  - `int matrix_size`: Size of the square matrices for multiplication (set to 400).
  - `int result_matrix[400][400], matrix_A[400][400], matrix_B[400][400]`: Arrays to store the result matrix and the two input matrices.
- Thread Function
  - `void *matrixWorker(void *thread_ptr)`: This function is executed by each pthread. It computes a subset of rows of the result matrix multiplication.
  - It uses a semaphore to signal when each thread completes its computation.
- Main Function
  - Parses the command-line argument to determine the number of threads to be used.
  - Initializes the semaphore using `sem_init()`.
  - Initializes the input matrices `matrix_A` and `matrix_B` with values of 1.
  - Measures the start time using `etime()`.
  - Creates `num_threads` pthreads, each executing the `matrixWorker` function.
  - Waits for all threads to complete using `sem_wait()` on the semaphore.
  - Measures the elapsed time again using `etime()` after all threads complete their computation.
  - Prints the elapsed time.

In summary, this program performs matrix multiplication in parallel using pthreads. It divides the task among multiple threads, synchronizes them using a semaphore, and measures the time taken for the entire multiplication process to complete. Finally, it prints the elapsed time.

## Project 1 Report

### Output of para\_mm.c and etime.c



```
shhonnag@spirit:~/proj1$ ./para_mm 1
Elapsed time: 0.305566
shhonnag@spirit:~/proj1$ ./para_mm 2
Elapsed time: 0.156265
shhonnag@spirit:~/proj1$ ./para_mm 4
Elapsed time: 0.080938
shhonnag@spirit:~/proj1$ ./para_mm 8
Elapsed time: 0.080968
shhonnag@spirit:~/proj1$ ./para_mm 16
Elapsed time: 0.082069
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
shhonnag@spirit:~/proj1$
```

```
shhonnag@spirit:~/proj1$ ./para_mm 1
Elapsed time: 0.305566
shhonnag@spirit:~/proj1$ ./para_mm 2
Elapsed time: 0.156265
shhonnag@spirit:~/proj1$ ./para_mm 4
Elapsed time: 0.080938
shhonnag@spirit:~/proj1$ ./para_mm 8
Elapsed time: 0.080968
shhonnag@spirit:~/proj1$ ./para_mm 16
Elapsed time: 0.082069
```

### makefile

- Variables
  - proj1 = gcc: Assigns the value gcc to the variable proj1, which represents the C compiler to be used.
  - LIBS = -lpthread: Specifies a library (-lpthread) to be linked during compilation.
- Targets
  - all: This target indicates that when you just run make, it should build all the targets listed after it (et, ep, para\_mm).
  - et, ep, para\_mm: These are the targets for building individual executables. Each target depends on corresponding object files and etime.o. When make is called with any of these targets, it will build the specified executable using the associated object files.



# Project 1 Report

- Dependencies
  - For each of the executables (et, ep, para\_mm), there are dependencies listed:
  - For example, et: et.o etime.o means that the et target depends on et.o and etime.o. If any of these files change, et needs to be rebuilt.
- Compilation Commands
  - The lines starting with et:, ep:, para\_mm:, et.o:, ep.o:, para\_mm.o:, etime.o: are rules that tell make how to build the targets or object files listed after the colon.
  - Each rule specifies the compilation command to compile the corresponding source file into an object file. For example, \$(proj1) -c et.c means to compile et.c using the C compiler specified by proj1 into et.o.
- Clean Target
  - clean: This target specifies how to clean up the project. It typically removes any generated files. In this case, it removes all object files (\*.o) and executables (et, ep, para\_mm).

In summary, this makefile automates the compilation process for a project containing multiple C source files. It compiles each source file into an object file and then links them together to create the final executables `et`, `ep`, and `para mm`.

The screenshot displays a MobaXterm window titled "spirit.eecs.csuohio.edu". The interface includes a top menu bar with options like Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, and Help. On the left, there's a sidebar with icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help. Below this is a "Quick connect..." field and a file explorer showing the directory structure of the remote host.

The main terminal pane shows the following commands and output:

```
shhonnag@spirit:~/proj1$ make all
gcc -c et.c
gcc -o et et.o etime.o -lpthread
gcc -c ep.c
gcc -o ep ep.o etime.o -lpthread
gcc -c para_mm.c
gcc -o para_mm para_mm.o etime.o -lpthread
shhonnag@spirit:~/proj1$ ssh grail
shhonnag@grail's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-169-generic x86_64)

** System restart required **

*****
This machine for educational purposes only. This does not include GAMES
or HACKING/SPAMMING/ANNOYING OTHER SYSTEMS or ANY OTHER NON-SCHOOL
RELATED ACTIVITIES. Anyone found participating in such activities will
LOSE their account (this will possibly lead to failure in some classes).
All activities on this machine may be monitored. Any illegal and/or
destructive activities will be reported to the proper authorities.
*****

Please don't run any hw/proj programs on grail. Use ssh to access other
LINUX machines spirit, arthur, bach, chopin, davinci, etc.

*****
Last login: Tue Feb 13 22:38:35 2024 from 137.148.204.14
shhonnag@grail:~$ cd proj1
shhonnag@grail:~/proj1$ ls
ep  ep.c  ep.o  et  et.c  etime.c  etime.o  et.o  makefile  para_mm  para_mm.c  para_mm.o  report.pdf
```

At the bottom of the terminal, there are several prompts for the user to enter commands, indicating they are still in the directory ~/proj1.

The status bar at the very bottom of the window indicates it is an "UNREGISTERED VERSION" and provides a link to support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>. It also shows system information like temperature (0°C), search bar, taskbar icons, and system clock (17:00, 14-02-2024).

## Testing and Debugging Experience

Testing and debugging were vital across all project components.

Debugging `etime.c` ensured precise time measurement synchronization for all other c code (`ep.c`, `et.c` and `para_mm.c`).

For `ep` and `et.c` various buffer sizes and memory operations were checked, while debugging resolved segmentation faults and ensured robust command-line argument handling. It was also focused on diverse thread counts and memory operation combinations, with debugging addressing synchronization challenges.

For `para_mm.c`, I made sure matrix multiplication was happening the way it should and also made sure semaphore sync was implemented and used properly.

For `makefile` I made sure it had correct dependency handling, with debugging efforts aimed at resolving compilation errors while ensuring compatibility with `etime.c`.

### PROJECT STATUS :

<code>etime.c</code>	Working as expected
<code>ep.c</code>	Working as expected
<code>et.c</code>	Working as expected
<code>para_mm.c</code>	Working as expected
<code>ep.c</code> and <code>etime.c</code>	Working as expected
<code>et.c</code> and <code>etime.c</code>	Working as expected
<code>para_mm.c</code> and <code>etime.c</code>	Working as expected

So on overall project status is “working successfully”.

## Experimental Results

`ep.c` and `etime.c`

Size	-b	-a
10	0.000622	0.000760
100	0.000404	0.000760
1000	0.000203	0.002932
10000	0.000491	0.006235
100000	0.002580	0.054183

`et.c` and `etime.c`

Size	-b	-a
10	0.000261	0.000143
100	0.000298	0.000077

## Project 1 Report

1000	0.000329	0.000274
10000	0.000161	0.000144
100000	0.000509	0.000129

para\_mm.c and etime.c

No. of threads	elapsed time
1	0.295935
2	0.158833
3	0.136861
4	0.106552
5	0.081230

I have tried with different values as input have recorded the results.

And by following all the instructions given by professor I was able to complete the whole project successfully.