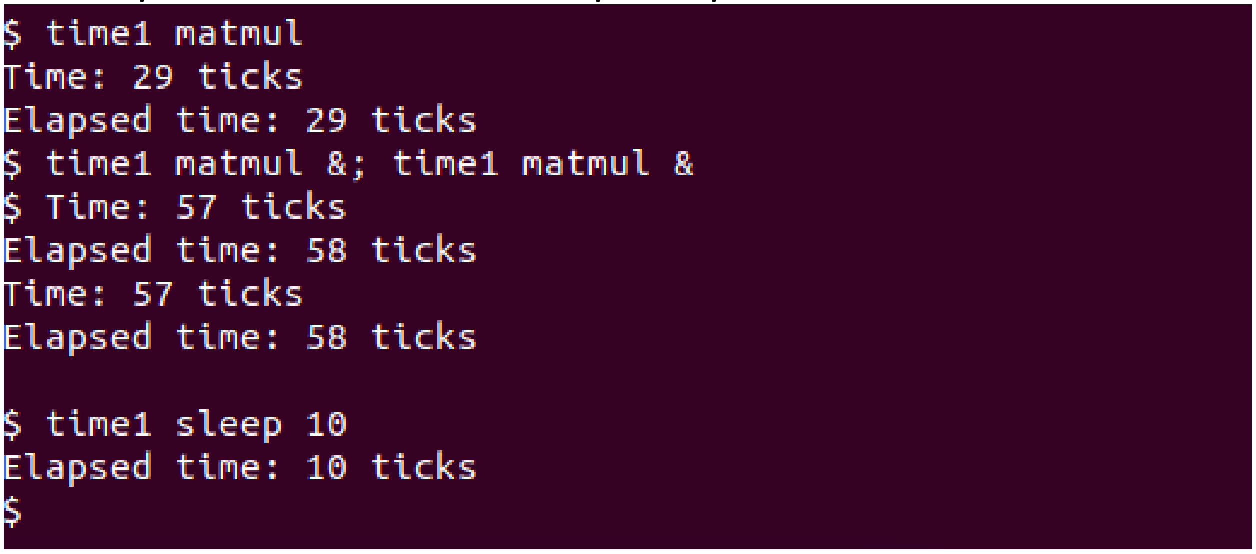
CS4375-13948 Fall 2023
<Javier Venegas>
<10/04/2023>
javenegas8@miners.utep.edu
https://github.com/javenegas8/OperatingSMoore

Homework 2 Report
Submitted on

## HW 2: Time Command

**Task 1. Implement a time1 command that reports elapsed time.**



```
$ time1 matmul
Time: 29 ticks
Elapsed time: 29 ticks
$ time1 matmul &; time1 matmul &
$ Time: 57 ticks
Elapsed time: 58 ticks
Time: 57 ticks
Elapsed time: 58 ticks

$ time1 sleep 10
Elapsed time: 10 ticks
$
```

I learned how to handle command-line arguments passed to a program, which is a fundamental skill for building command-line utilities. Creating a complete program that solves a specific problem, measuring the execution time of other programs in this case helps improve your problem-solving skills and programming abilities.

Describe any difficulties you ran into with this task and if/how you overcame them.

Some of the difficulties that become a challenge in this task is understand how the commands work perfectly and revising the CPU to run both programs time1 and matmul. I dedicated some time in research to develop a better plan and outcome for the xv6 to work.

**Task 2. Keep track of how much cputime a process has used.**

The files changed in the repository were kernel/proc.h. adding the cputime field to the struct command. Initialize the field to zero when the process is created in allocproc() in kernel/proc.c. Increment the field every time the process uses up its time slice.

Summarize what you learned by carrying out this task.

I learned to work with the CPU of Ubuntu and revise the tasks that are made on the background of the system to keep track of the cputime.

Describe any difficulties you ran into with this task and if/how you overcame them.

The difficulties proceeding to create this task was to understand completely what the right commands for the specific files would be, this help me develop knowledge to modify the cputime correctly.

**Task 3. Implement a wait2() system call that waits for a child to exit and returns the child's status and rusage.**

List the files you changed and explain the purpose of each change. Be sure to do git add and git

The files changed in this task were kernel/pstat.h by adding the struct rusage command, Create a system call with the following prototype (put this in user/user.h):int wait2(int*, struct rusage*);

•Add an entry for wait2() in user/usys.pl•

Add wait2 system call information to kernel/syscall.h and kernel/syscall.c

Implemented sys_wait2() in kernel/sysproc.c and wait2() in kernel/proc.c

Summarize what you learned by carrying out this task.

I learned that the difficulties between a wait system is how can a child will exit and can return the actual status. By implementing a wait system in the cputime the task became more precise returning the status and rusage.

Describe any difficulties you ran into with this task and if/how you overcame them.

The difficulties seen in this task was how to create and implement the system without destructing the system and the files already compiled. To work on this, I develop a algorithm that returns the havekids integer to get the status of the cputime

**Task 4. Implement a time command that runs the command given to it as an argument and outputs elapsed time, CPU time, and %CPU used.**

Your time.c code should be in the user directory in the hw2 branch of your xv6 repo. Remember to add your new command to UPROGS in Makefile.

Show results from running your time command.

**Extra Credit (5 points). Discuss limitations of our time command. (Hint: For one limitation, consider what would happen if the command that is being timed forks child processes).**

One significant limitation is Inability to track child processes, it cannot accurately track the execution time of commands that fork child processes. When a command forks child processes, the time1 command only measures the time taken by the parent process, not the cumulative time of all child processes. This can lead to inaccurate timing results for commands that heavily rely on parallelism or multiple subprocesses.