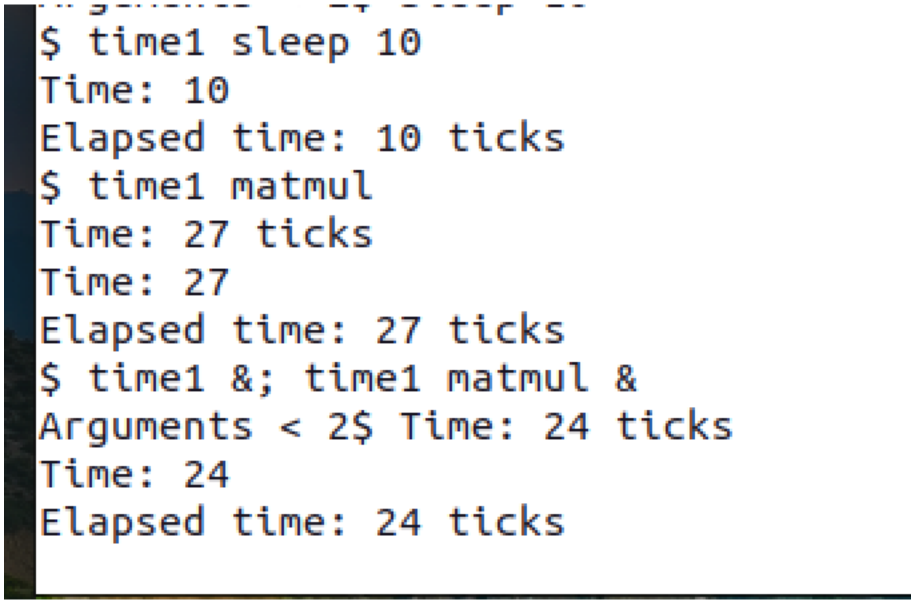
CS4375-13948 Fall 2023 Homework Report
Geoff Waller Submitted on 10/4/23
gwaller@miners.utep.edu

HW 2: Time Command

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

Time1 results:



```
$ time1 sleep 10
Time: 10
Elapsed time: 10 ticks
$ time1 matmul
Time: 27 ticks
Time: 27
Elapsed time: 27 ticks
$ time1 &; time1 matmul &
Arguments < 2$ Time: 24 ticks
Time: 24
Elapsed time: 24 ticks
```

With this task, I learned how to implement a system call. I learned how to assemble arguments for an exec call, as using a the uptime command to get elapsed time for a system call.
I ran into difficulties in how I assemble the exec() call at first. I did not fully understand that I had to move arguments to the right(or higher number).

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

For task 2, I added a cputime field to struct proc which I set to 0, In trap.c, I incremented cputime in both the usertrap() and kernaltrap() functions. For this task I learned how to better trace what my programs and functions are doing through multiple files.

For this task, I did not have any difficulties.

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

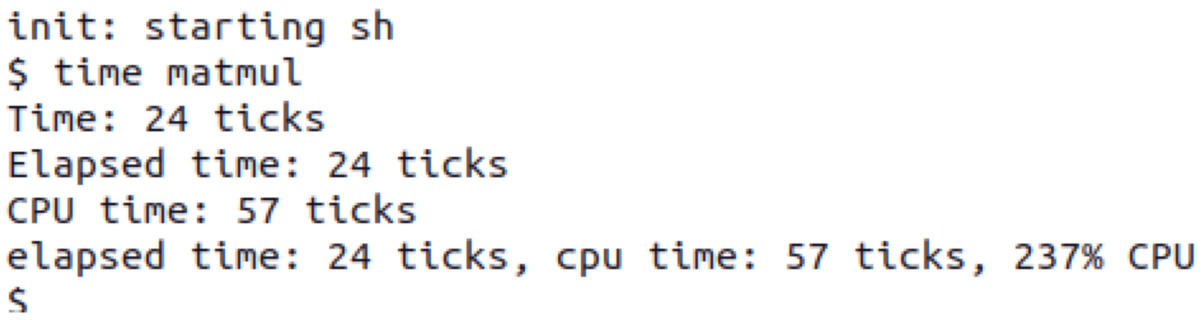
In task 3, pstat.h was edited to allow a struct rusage to be a pointer for cputime. Struct rusage was then added in user.h. This enables the pointer to rsuage to be added to a new system call (wait2) that takes in two parameters, the second being a pointer to rsuage. Wait2 was then added to usys.pl to enable the system to add it as a system call for the user and same in the kernel in both syscall.h and syscall.c. A function for sys_wait2 was added in sysproc.c in order turn get the system to return the process and rusuage (the cputime.) This was done in proc.c for the same reason.

This proved to be the most difficult task for me. I learned more about how the system architecture works. I had to be care to trace each function and variable carefully through several files in both the kernel and user side in order to make sense of what the code was doing. I also learned I need to make changes to both user and kernel side in order to get the code to work properly. I have a better understanding of how the flow of the system but I still get lost.

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.

My time.c is not currently 100% correct. I ran out of time to get the algorithm for the cpu usage correct. However, I fell with a little more time I could get the output correct.

Time.c results:



```
init: starting sh
$ time matmul
Time: 24 ticks
Elapsed time: 24 ticks
CPU time: 57 ticks
elapsed time: 24 ticks, cpu time: 57 ticks, 237% CPU
$
```