Lab: Timing Code in C

How to measure runtime in serial C code.



Login to "cms-grid" multicore nodes.

PuTTY Configuration		? ×	
Category:			
Session Logging Terminal Keyboard Bell Features Window Appearance Behaviour Translation Selection Colours Connection Data Proxy Telnet Rlogin SSH Serial	Basic options for your PuTTY session		
	Specify the destination you want to connect to Host Name (or IP address) cms-grid-login.cms.gre.ac.uk Connection type: Raw Telnet Rlogin SSH Load, save or delete a stored session Saved Sessions	Port 22	
	Default Settings	Load Save Delete	
	Close window on exit: Always Never Only on clean exit		
About Hel	p <u>O</u> pen	<u>C</u> ancel	

₫ tc70@cms-grid-06:~	_	×
login as: staff\tc70 Pre-authentication banner message from server:		^
Unauthorized use of University of Greenwich computers and netwo resources is prohibited. If you log on to this computer syst acknowledge your awareness of and concurrence with the Uni of Greenwich personal conduct code and JANET acceptable use pol	em, yo versit	
End of banner message from server staff\tc70@cms-grid-login.cms.gre.ac.uk's password: Last login: Thu Sep 24 13:44:12 2020 from 193.60.76.221 [tc70@cms-grid-06 ~]\$		



```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
int main(int argc, char** argv)
    //The variables for the start and stop timer
    struct timeval startTime, stopTime;
    //loop variable
    int i;
    // This variable will hold the total time
    long totalTime;
    // Start timer: get current time and store it in variable startTime
    gettimeofday(&startTime, NULL);
    for (i = 0; i < 10000; i++) {
    printf("hello world\n");
    // Stop timer: get current time and store it in variable stopTime
    gettimeofday(&stopTime, NULL);
    // Calculate total time by subracting the startTime from the stopTime (result is in microseconds)
    totalTime = (stopTime.tv sec * 1000000 + stopTime.tv usec) - (startTime.tv sec * 1000000 + startTime.tv usec);
    // Print the totalTime as a long integer (%ld)
    printf("%ld\n", totalTime);
    return (0);
```

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

int main(int argc, char** argv)
{
    //The variables for the start and stop timer
    struct timeval startTime, stopTime;
```



```
int main(int argc, char** argv)
    //The variables for the start and stop timer
    struct timeval startTime, stopTime;
    //loop variable
    int i;
    // This variable will hold the total time
    long totalTime;
                         struct timeval {
                             time_t tv_sec; /* seconds */
                             suseconds_t tv_usec; /* microseconds */
                         };
```



```
// Start timer: get current time and store it in variable startTime
  gettimeofday(&startTime, NULL);
  for (i = 0; i < 10000; i++) {
  printf("hello world\n");
  // Stop timer: get current time and store it in variable stopTime
  gettimeofday(&stopTime, NULL);
// Calculate total time by subracting the startTime from the stopTime (result is in microseconds)
totalTime = (stopTime.tv sec * 1000000 + stopTime.tv usec) - (startTime.tv sec * 1000000 + startTime.tv usec);
// Print the totalTime as a long integer (%ld)
printf("%ld\n", totalTime);
```



Timing code in C

- Download helloWorld.c from moodle upload to the unix system compile and run it
- 2. Do the same for measureTime.c and sumInt.c
- 3. Modify sumInt.c to add timings
- 4. Increase the loop length to 1 million. Time the code again
- 5. Run timings in incremements of 100,000 loops starting at 100,000 and finishing at 1 million. How does the time increase?

