

به نام خدا

گزارش پروژه ی ۱ سیستم عامل

۲- در سیستم عامل ویندوز، برنامه تک‌نخی (single thread) خواسته‌شده را بنویسید و آن را به ازای ورودی‌های خواسته‌شده سه‌بار اجرا کنید. میانگین زمان اجرا، حافظه مصرفی و بهره‌وری پردازنده^۱ را گزارش کنید.

۳- برنامه خود را در حالت‌های زیر مجدداً اجرا کنید. مجدداً میانگین زمان اجرا، حافظه مصرفی و بهره‌وری پردازنده را گزارش کنید.

الف) با استفاده از چندین فرآیند (پردازه)^۲

ب) با استفاده از چندین ریسمان (نخ)

۴- مراحل ۲ و ۳ را در مجدداً در لینوکس پیاده‌سازی کنید.

۵- تحلیل خود از نتایج بدست آمده در قالب جدول بنویسید. علت تفاوت‌ها در مقادیر بدست آمده را بحث کنید؟

:مساله ی مورد نظر

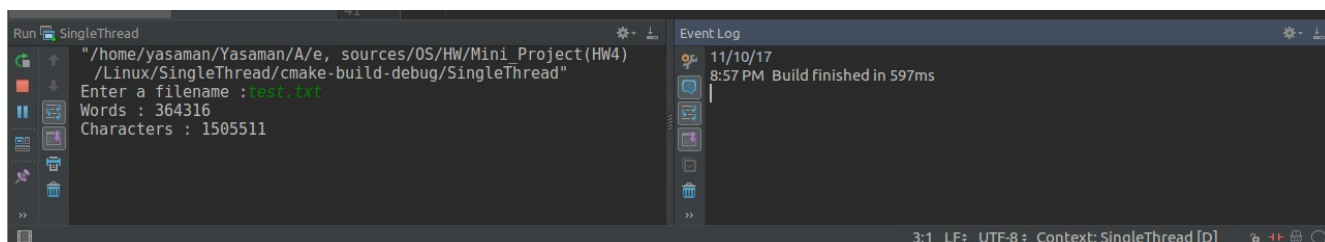
*شمارش کلمات داخل یک متن (حداقل ۱ میلیون کاراکتر) *

زبان مورد استفاده
C

یاده سازی در لینوکس ۱)

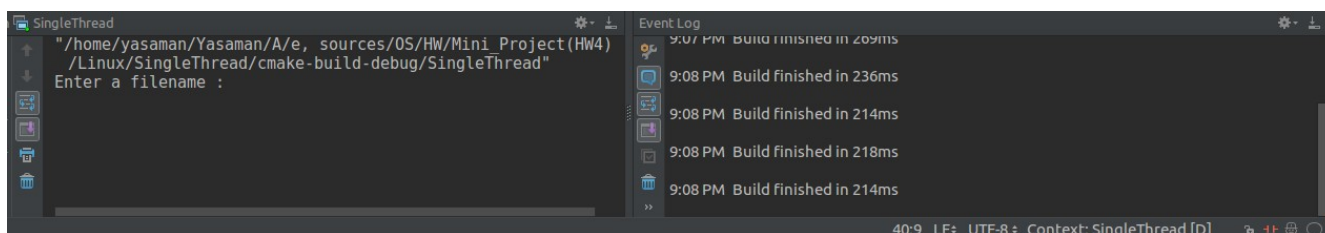
Single Thread

میانگین زمان اجرا



```
Run SingleThread
"/home/yasaman/Yasaman/A/e, sources/OS/HW/Mini Project(HW4)
/Linux/SingleThread/cmake-build-debug/SingleThread"
Enter a filename :test.txt
Words : 364316
Characters : 1505511

Event Log
11/10/17
8:57 PM Build finished in 597ms
```

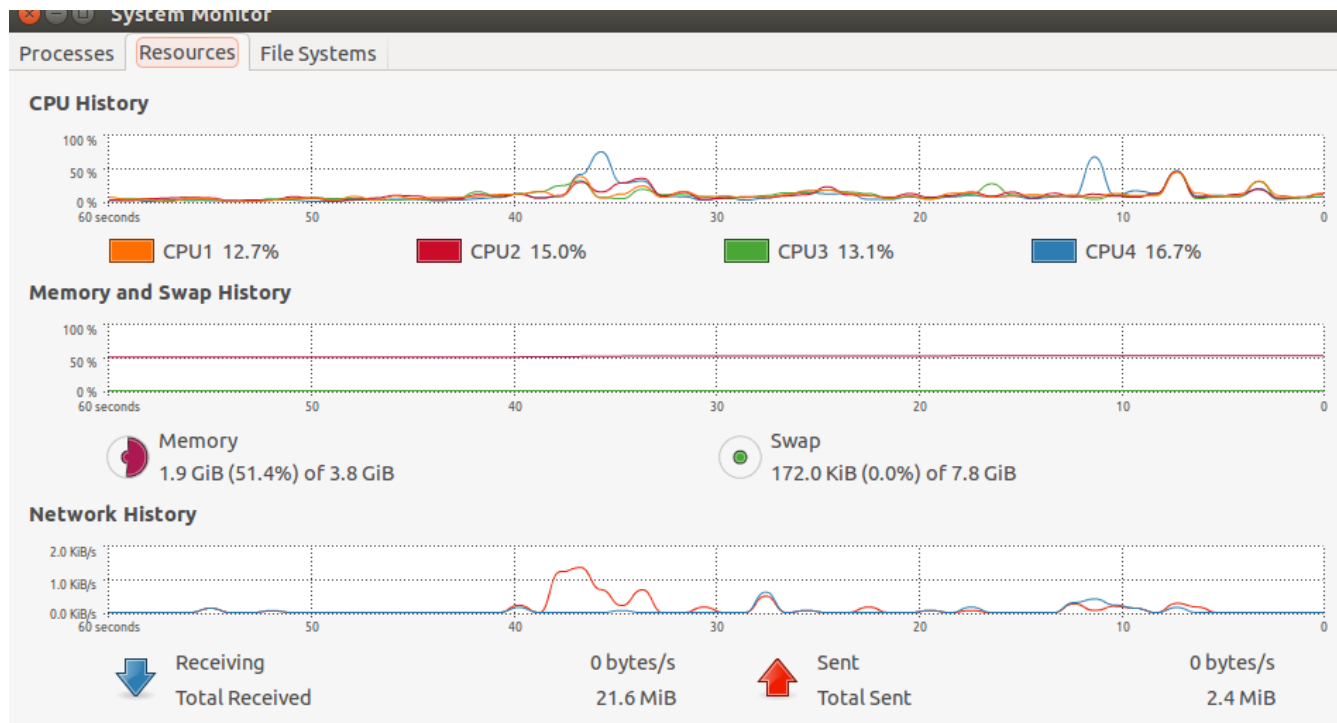


```
SingleThread
"/home/yasaman/Yasaman/A/e, sources/OS/HW/Mini Project(HW4)
/Linux/SingleThread/cmake-build-debug/SingleThread"
Enter a filename :

Event Log
9:07 PM Build finished in 209ms
9:08 PM Build finished in 236ms
9:08 PM Build finished in 214ms
9:08 PM Build finished in 218ms
9:08 PM Build finished in 214ms
```

مصرف حافظه و بهره وری پردازنده

init	root	0	1	3.2 MiB	Normal
irq/62-mej_me	root	0	398	N/A	Normal
irq/64-iwlwifi	root	0	414	N/A	Normal
irqbalance	root	0	1212	772.0 K	Normal
java	yasamar	0	2964	632.3 K	Normal
jbd2/sda4-8	root	0	201	N/A	Normal
kauditd	root	0	670	N/A	Normal
kblockd	root	0	41	N/A	Very High



Multi Thread

Why Multithreading?

Threads are popular way to improve application through parallelism. For example, in a browser, multiple tabs can be different threads. MS word uses multiple threads, one thread to format the text, other thread to process inputs, etc.

Threads operate faster than processes due to following reasons:

- 1) Thread creation is much faster.
- 2) Context switching between threads is much faster.
- 3) Threads can be terminated easily
- 4) Communication between threads is faster.

Pseudo code:

```
def main:
    size = filesize
    ptr = mmap(file)
    num_threads = 4

    for i in range(1, num_threads):
        new_thread(exec = count_words,
                    start = ptr + i * size / num_threads,
                    length = size / num_threads)
```

```
wait_for_result(all_threads)
join_the_results
```

```
def count_words(start, length):
    # Count words as if it was an entire file
    # But store separately the first/last word if
    # the segment does not start/ends with an word
    # separator(" ", ".", ",", "\n", etc...)
    return (count_of_words, first_word, last_word)
```

مصرف حافظه و بهره‌وری پردازنده
در این حالت زمان اجرا بسیار کمتر از حالت قبلی خواهد بود (حدود ۱۰۰ میلی ثانیه - میانگین)
(به طرز قابل توجهی)

مصرف حافظه نیز بسیار کمتر خواهد بود (چون از مقدار مشخصی حافظه به طور مشترک استفاده میکند تا یک برنامه را
ب صورت تکه تکه اجرا نماید)

Multi Process

باید از تکنیک های حافظه مشترک استفاده کنیم

توضیح

لازم به ذکر است چون سیستم عامل لینوکس با تمام فرآیندها تحت عنوان "تسک" برخورد میکند و کانتکست سوئیچ های
انجام شده روی هر دو به یک سیوه انجام میشود در نتیجه تفاوت سرعت یا مصرف حافظه و بهره‌وری پردازنده به طور قابل
ملاحظه در اینجا دیده نمیشود

به عبارتی در لینوکس مرز بین "نخ" و "فرآیند" بسیار کم رنگ است و تاثیر خاصی در نتیجه ندارد +

پیاده سازی در ویندوز (2)

در حالت تک نخ تفاوتی ایجاد نمیشود
در دو حالت بعدی نیز در مجموع مدیریت حافظه
و بهره‌وری سی پی یو و زمان اجرا

.

(در ویندوز مالتی پراسسینگ بهتر از مالتی ترد عمل میکند)

سیستم عامل لینوکس بهتر عمل میکند

Multithreading vs multiprocessing

Multithreading means exactly that, running multiple threads. This can be done on a uni-processor system, or on a multi-processor system.

On a single-processor system, when running multiple threads, the actual observation of the computer doing multiple things at the same time (i.e., multi-tasking) is an illusion, because what's really happening under the hood is that there is a software scheduler performing time-slicing on the single CPU. So only a single task is happening at any given time, but the scheduler is switching between tasks fast enough so that you never notice that there are multiple processes, threads, etc., contending for the same CPU resource.

On a multi-processor system, the need for time-slicing is reduced. The time-slicing effect is still there, because a modern OS could have hundred's of threads contending for two or more processors, and there is typically never a 1-to-1 relationship in the number of threads to the number of processing cores available. So at some point, a thread will have to stop and another thread starts on a CPU that the two threads are sharing. This is again handled by the OS's scheduler. That being said, with a multiprocessors system, you *can* have two things happening at the same time, unlike with the uni-processor system.

In the end, the two paradigms are really somewhat orthogonal in the sense that you will need multithreading whenever you want to have two or more tasks running asynchronously, but because of time-slicing, you do not necessarily need a multi-processor system to accomplish that. If you are trying to run multiple threads, and are doing a task that is highly parallel (i.e., trying to solve an integral), then yes, the more cores you can throw at a problem, the better. You won't necessarily need a 1-to-1 relationship between threads and processing cores, but at the same time, you don't want to spin off so many threads that you end up with tons of idle threads because they must wait to be scheduled on one of the available CPU cores. On the other hand, if your parallel tasks requires some sequential component, i.e., a thread will be waiting for the result from another thread before it can continue, then you may be able to run more threads with some type of barrier or synchronization method so that the threads that need to be idle are not spinning away using CPU time, and only the threads that need to run are contending for CPU resources.

(from stackoverflow)

در پایان مشاهده ی این لینک ها نیز خالی از لطف برای من نبود

*https://www.slideshare.net/Krishna972076/final-ppt-32222803?next_slideshow=1

*<https://ubuntuforums.org/showthread.php?t=811144>

در نهایت روشن است که استفاده از مالتی پروسسینگ و مالتی تردینگ در هر صورت بهتر از حالت تک نخه است

اما در مورد اینکه کدامیک بهترندُ به کاربرد و نوع برنامه بستگی دارد