### **Задание 1**

Многозадачность в современных системах реализована через переключение между процессами с помощью планировщика. Какие операции при этом замедляют работу компьютера?

*Приведите ответ в свободной форме со своим комментарием.*

**Ответ:** Замедляют - context-switch - это процесс сохранения с процессора рабочих регистров в отдельную память, когда процесс перемещается с процессора в очередь и потом загрузка из памяти в процессор - данные в регистры, когда процесс продолжает работать далее - помещается на процессор из очереди планировщиком. Чем чем меньше квант времени работы на процессоре - тем чаще и больше по времени в % соотношении будут контекст свитчи. Если время увеличить, то, меньше % времени будет тратиться на контекст-свитчи, но, может быть долгий отклик, т.е. прцоесс как бы не сразу отреагирует на изменения, т.к. не сразу попадет на процессор.

### **Задание 2**

На лекции приводились примеры использования планировщика SCHED\_DEADLINE: реагирование на датчики АЭС, обработка звука в реальном времени. В каких еще случаях можно использовать этот планировщик?

*Приведите ответ в свободной форме со своим комментарием.*

**Ответ:**

Что-то на лекции не приводились примеры.

SCHED\_DEADLINE используется для задач, чувствительных к задержке и времени завершения. Предполагаю, что его полезно использовать например где нужно точно по времени фиксировать например сигналы. Например, может быть, какой-то программный осциллограф, который получает сигналы по порту USB или еще как-то.

### **Задание 3**

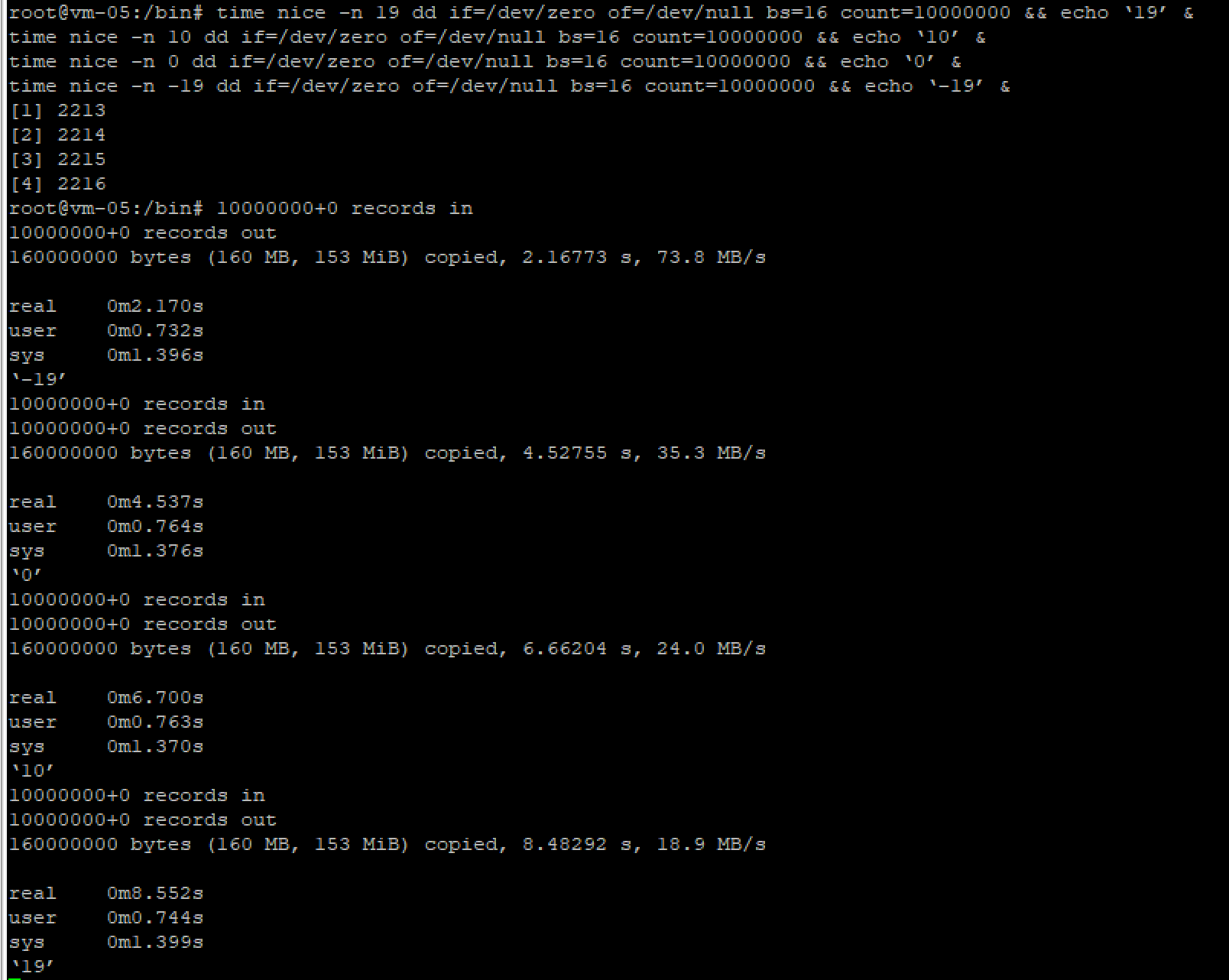
Запустите следующий код, имитирующий нагрузку типа ввод / вывод, с значениями nice 19, 10, 0 и измерьте время исполнения с помощью утилиты time. Объясните получившееся различие во времени исполнения для разных запусков.

dd if=/dev/zero of=/dev/null bs=16 count=10000000

*Ответ приведите в виде снимка экрана с комментариями в свободной форме*

**Решение:**

Я изменил код, сделал в текстовом редакторе заранее несколько строчек кода и одновременно разом все запустил:



Запускал одновременно все, чтобы видно было бы как работал nice .

Как и ожидалось, быстрее всех отработал процесс с nice -19 и т.д. в порядке увеличения nice.

Как я предполагаю:

real - это полное время работы.

user - это время работы самого процесса пользователя, т.е. кода процесса на процессоре.

sys - это видимо время работы планировщика - накладные расходы, чтобы работал пользовательский процесс на процессоре - ушло на контекст-свитчи?

если отнять real - sys - user = то видимо получится время, сколько процесс ожидал в очереди работы на процессоре ?

Верно ли?

### **Задание 4**

Повлияет ли на реальное время исполнения запуск кода из Задания 3 на одном ядре вместо нескольких? Напишите почему да или почему нет. Проверьте своё предположение с помощью утилиты taskset

*Ответ приведите в виде снимка экрана с комментариями в свободной форме*

**Ответ:**

Должно повлиять. Чем больше процессоров (ядер), тем больше параллельно сможет выполняться процессов. И тем быстрее выполнятся процессы. в прошлый раз у ВМ было 1 ядро (в п.3). Добавил 4 ядра, еще раз запустил, результат:

root@vm-05:~# time nice -n 19 dd if=/dev/zero of=/dev/null bs=16 count=10000000 && echo ‘19’ &

time nice -n 10 dd if=/dev/zero of=/dev/null bs=16 count=10000000 && echo ‘10’ &

time nice -n 0 dd if=/dev/zero of=/dev/null bs=16 count=10000000 && echo ‘0’ &

time nice -n -19 dd if=/dev/zero of=/dev/null bs=16 count=10000000 && echo ‘-19’ &

[1] 1107

[2] 1108

[3] 1110

[4] 1112

root@vm-05:~# 10000000+0 records in

10000000+0 records out

160000000 bytes (160 MB, 153 MiB) copied, 6.2254 s, 25.7 MB/s

real 0m6.228s

user 0m2.412s

sys 0m3.815s

‘-19’

10000000+0 records in

10000000+0 records out

160000000 bytes (160 MB, 153 MiB) copied, 6.30026 s, 25.4 MB/s

real 0m6.303s

user 0m2.311s

sys 0m3.989s

‘19’

10000000+0 records in

10000000+0 records out

160000000 bytes (160 MB, 153 MiB) copied, 6.31729 s, 25.3 MB/s

real 0m6.320s

user 0m2.336s

sys 0m3.978s

‘10’

10000000+0 records in

10000000+0 records out

160000000 bytes (160 MB, 153 MiB) copied, 6.45366 s, 24.8 MB/s

real 0m6.456s

user 0m2.215s

sys 0m4.237s

‘0’

Странно, что все стали работать долго - около 6 сек. Не странно, что не в порядке приоритета длительность, т.к. конкуренции не было - каждый процесс попал на свое ядро и параллельно выполнился. Вот почему стало дольше - у всех около 6 сек - я не понял. Почему?

### **Задание 5\***

При каких условиях лучше увеличить time slice планирования, а когда лучше уменьшить?

*Приведите ответ в свободной форме со своим комментарием.*

Если нам не важно время отклюика на внешние события но важно максимально эффективно использовать мощности CPU то, лучше увеличить время time slice. Если же важно быстрый отклик процесса на события, и есть запас мощностей CPU и нам их не жалко, то, можно уменьшить время.