Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

Рассчётно-графичеческая работа  
по дисциплине “Операционные системы реального времени”

Выполнил:

студент группы

Работу проверил:

Заведующий кафедрой ПМиК

Фионов А.Н.

# Задание

1. Сравните эффективность по времени реализации критических секций с помощью семафора и мьютекса.
2. Аппаратная поддержка механизмов управления памятью может быть различной. Действует ли флаг PROT\_NOCACHE при выделении страницы памяти с помощью mmap()?

# Идея постановки эксперимента

1. Необходимо создать задачу с использованием большого числа потоков и сравнить эффективность двух блокировщиков: семафора и мьютекса
2. Необходимо получить доступ к странице выделенной памяти при помощи mmap(). Сравнить результаты от использования флага PROT\_NOCACHE.

# Текст программы

## mutex.cpp

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #include <inttypes.h>  #include <sys/neutrino.h>  static int counter = 0;  pthread\_mutex\_t mutex;  void\* plus(void \*args)  {  int local;  pthread\_mutex\_lock(&mutex);  local = counter;  //printf("Plus: %d\n", counter);  local++;  counter = local;  pthread\_mutex\_unlock(&mutex);  }  void\* minus(void \*args)  {  int local;  pthread\_mutex\_lock(&mutex);  local = counter;  printf("Minus: %d\n", counter);  local--;  counter = local;  pthread\_mutex\_unlock(&mutex);  }  #define NUM\_OF\_THREADS 50000  int main()  {  uint64\_t start, finish;  pthread\_t threads[NUM\_OF\_THREADS];  start = ClockCycles();  printf("Counter start state: %d\n", counter);  pthread\_mutex\_init(&mutex, NULL);  for(size\_t i = 0; i < NUM\_OF\_THREADS/2; i++)  pthread\_create(&threads[i], NULL, minus, NULL);  for(size\_t i = NUM\_OF\_THREADS/2; i < NUM\_OF\_THREADS; i++)  pthread\_create(&threads[i], NULL, plus, NULL);  for(size\_t i = 0; i < NUM\_OF\_THREADS; i++)  pthread\_join(threads[i], NULL);  pthread\_mutex\_destroy(&mutex);  printf("Counter finish state: %d\n", counter);  finish = ClockCycles();  printf("Clocks: %u\n", finish - start);  return 0;  } |

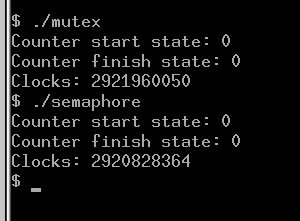
## semaphore.cpp

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>  #include <inttypes.h>  #include <semaphore.h>  #include <sys/neutrino.h>  static int counter = 0;  sem\_t semaphore;  void\* plus(void \*args)  {  int local;  sem\_wait(&semaphore);  local = counter;  //printf("Plus: %d\n", counter);  local++;  counter = local;  sem\_post(&semaphore);  }  void\* minus(void \*args)  {  int local;  sem\_wait(&semaphore);  local = counter;  printf("Minus: %d\n", counter);  local--;  counter = local;  sem\_post(&semaphore);  }  #define NUM\_OF\_THREADS 50000  int main()  {  uint64\_t start, finish;  pthread\_t threads[NUM\_OF\_THREADS];  start = ClockCycles();  printf("Counter start state: %d\n", counter);  sem\_init(&semaphore, 0, 1);  for(size\_t i = 0; i < NUM\_OF\_THREADS/2; i++)  pthread\_create(&threads[i], NULL, minus, NULL);  for(size\_t i = NUM\_OF\_THREADS/2; i < NUM\_OF\_THREADS; i++)  pthread\_create(&threads[i], NULL, plus, NULL);  for(size\_t i = 0; i < NUM\_OF\_THREADS; i++)  pthread\_join(threads[i], NULL);  sem\_destroy(&semaphore);  printf("Counter finish state: %d\n", counter);  finish = ClockCycles();  printf("Clocks: %u\n", finish - start);  return 0;  } |

## mman.cpp

|  |
| --- |
| #include <sys/neutrino.h>  #include <unistd.h>  #include <stdio.h>  #include <vingraph.h>  #include <time.h>  #include <pthread.h>  #include <sys/mman.h>  int main() {  int \*test = static\_cast<int\*> (  mmap(0, sizeof \*test,  PROT\_READ|PROT\_WRITE|PROT\_NOCACHE,  MAP\_SHARED|MAP\_ANONYMOUS,  -1, 0)  );  printf("%d\n", \*test);  \*test = 100;  printf("%d\n", \*test);  int \*test2 = static\_cast<int\*> (  mmap(0, sizeof \*test2,  PROT\_READ|PROT\_WRITE,  MAP\_SHARED|MAP\_ANONYMOUS,  -1, 0)  );  printf("%d\n", \*test2);  \*test2 = 100;  printf("%d\n", \*test2);  return 0;  } |

# Результаты

1. В теории мьютексы должны быть быстрее, но на практике результаты на уровне погрешности.  
   
2. Исходя из документации. флаг PROT\_NOCACHE следует использовать для предотвращения получения устаревшей, то есть закешированной памяти при получения доступа к памяти, которая может быть изменена аппаратно. На практике результатов обнаружено не было.  
   