

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

Расчетно-графическая работа по дисциплине
«Операционные системы реального времени»

Выполнил:
студент группы ИП-712
Рещиков А.Е.

Работу проверил:
Заведующий кафедрой ПМиК
Фионов А.Н.

Новосибирск 2020

Оглавление

Задание	3
Идея постановки эксперимента	4
Текст программы для задания № 1.1	5
with_signal.c	5
Текст программы для задания № 1.2	5
with_semaphore.c	5
Текст программы для задания № 2	6
mem_eater.c	6
Результаты	7
Используемые ресурсы	9

Задание

1. Сравните время активизации нити с помощью сигнала и семафора.
2. Установите, какой максимальный объем памяти может предоставить процессу система.

Идея постановки эксперимента

1. Необходимо реализовать программу для замера активизации нити с помощью сигнала/семафора.
Для этого были написаны 2 отдельные программы для сигнала и семафора, соответственно.
2. При поиске информации касательно второго задания документация QNX гласит, что максимальный размер памяти выделяемый под процесс имеет пределы от 0 до 3.5Gb

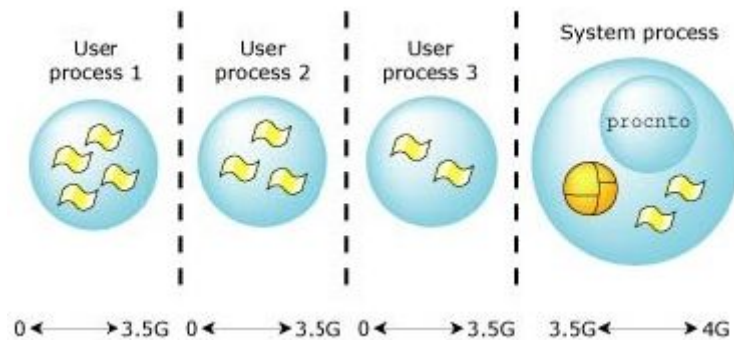


Рисунок из официальной документации QNX.

В ходе обдумывания простейшего из вариантов было аккумулировано решение о создании программы по выделению квадратичной матрицы больших размеров, поскольку максимальное число `int` все еще может быть спокойно выделено для массива.

Текст программы для задания № 1.1

with_signal.c

```
1. #include <stdio.h>
2. #include <time.h>
3. #include <pthread.h>
4. #include <signal.h>
5. #include <sys/neutrino.h>
6. #include <inttypes.h>
7. timespec start, end;
8. boolean flag = true;
9. uint64_t res, fres;
10. void *flag_off(){flag = false;}
11. void *thread(void *arg) {
12.     res = ClockCycles();
13.     clock_gettime(CLOCK_REALTIME, &start);
14.     while(flag){};
15. }
16. int main() {
17.     signal(0, SIG_IGN);
18.     pthread_create(0, 0, thread, 0);
19.     signal(6, flag_off);
20.     clock_gettime(CLOCK_REALTIME, &end);
21.     fres = ClockCycles();
22.     printf("Sec - %ld\n", end.tv_sec - start.tv_sec);
23.     printf("N-Sec - %ld\n", end.tv_nsec - start.tv_nsec);
24.     printf("CC - %ld\n", fres - res);
25.     return 0;}
```

Текст программы для задания № 1.2

with_semaphore.c

```
1. #include <stdio.h>
2. #include <time.h>
3. #include <pthread.h>
4. #include <semaphore.h>
5. #include <sys/neutrino.h>
6. #include <inttypes.h>
7. sem_t sem;
8. uint64_t res, fres;
9. timespec start, end;
10. void *thread(void *arg) {
11.     res = ClockCycles();
12.     clock_gettime(CLOCK_REALTIME, &start);
13.     sem_post(&sem);
14. }
15. int main() {
16.     sem_init(&sem, 0, 0);
17.     pthread_create(0, 0, thread, 0);
18.     sem_wait(&sem);
19.     fres = ClockCycles();
20.     clock_gettime(CLOCK_REALTIME, &end);
21.     printf("Sec - %ld\nN-sec - %ld\n", end.tv_sec - start.tv_sec, end.tv_nsec -
start.tv_nsec);
22.     printf("CC - %ld\n", fres - res);
23.     return 0;
24. }
```

Текст программы для задания № 2

mem_eater.c

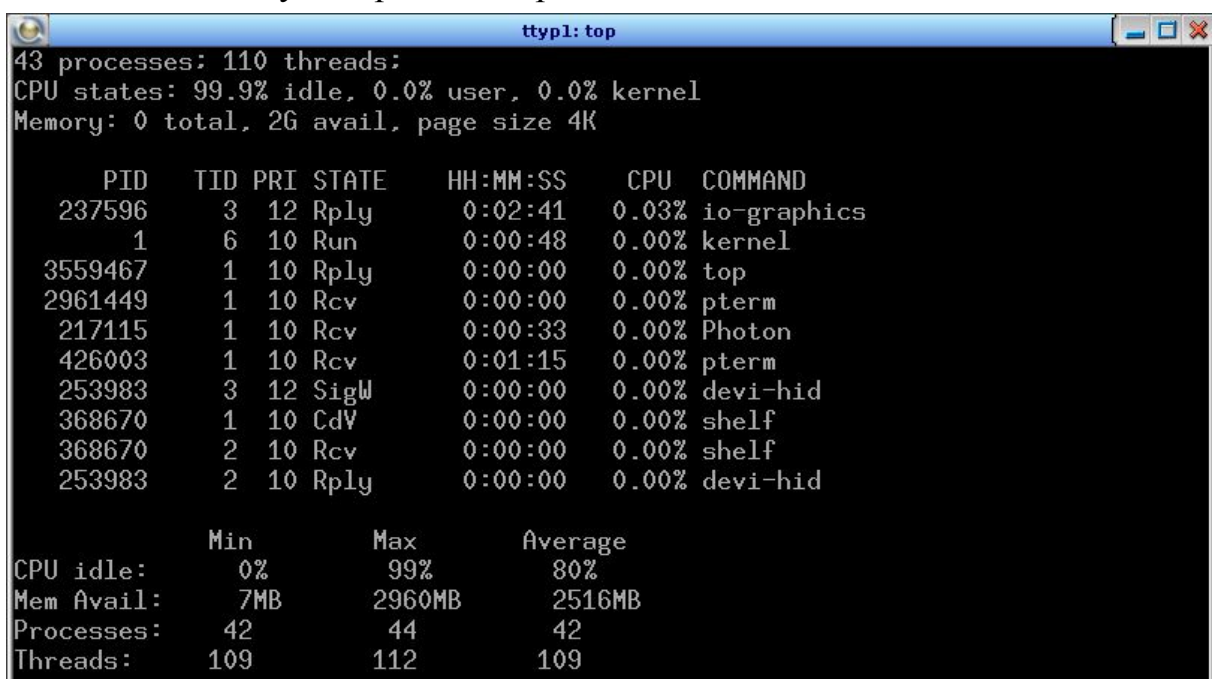
```
25. #include <stdio.h>
26. #include <stdlib.h>
27.
28. int main()
29. {
30.     int b = 100000;
31.     int **a;
32.     while(b > 0)
33.     {
34.         printf("%d\n", b);
35.         a = (int **) malloc(b * sizeof(int));
36.         for(int i = 0; i < b; i++) a[i] = (int *) malloc(b * sizeof(int));
37.         b = b + 100;
38.         printf("%d, b - %d\n", sizeof(a), b);
39.         free(a);
40.     }
41. }
```

Результаты

1. На практике было выявлено что сигналы немногим быстрее семафоров, однако, конкретно на моем домашнем ПК, времени на выделение было затрачено слишком мало, секунды и наносекунды просто не берутся, поэтому были применены циклы(*ClockCycles()*).

```
# ./sem
Sec - 0
N-sec - 0
CC - 275735
# ./sig
sig      signal.c
# ./sig
Sec - 0
N-Sec - 0
CC - 153759
#
```

2. При помощи top было выявлено максимальное количество памяти, которое я могу получить от ОС, при моей конфигурации это 2G, о чем свидетельствует скриншот top



```
43 processes: 110 threads:
CPU states: 99.9% idle, 0.0% user, 0.0% kernel
Memory: 0 total, 2G avail, page size 4K

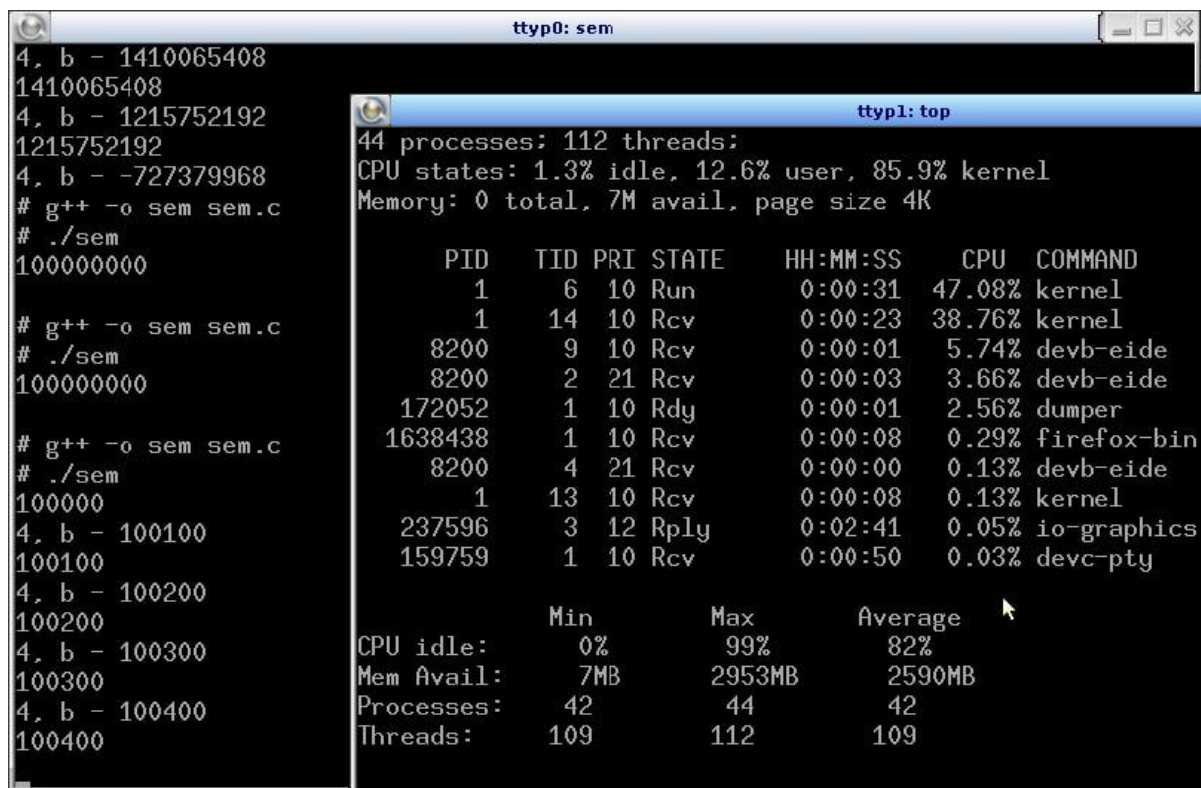
  PID    TID PRI STATE   HH:MM:SS   CPU  COMMAND
  237596   3  12 Rply    0:02:41   0.03% io-graphics
    1     6  10 Run     0:00:48   0.00% kernel
  3559467  1  10 Rply    0:00:00   0.00% top
  2961449  1  10 Rcv     0:00:00   0.00% pterm
  217115   1  10 Rcv     0:00:33   0.00% Photon
  426003   1  10 Rcv     0:01:15   0.00% pterm
  253983   3  12 SigW    0:00:00   0.00% devi-hid
  368670   1  10 CdV     0:00:00   0.00% shelf
  368670   2  10 Rcv     0:00:00   0.00% shelf
  253983   2  10 Rply    0:00:00   0.00% devi-hid

  CPU idle:      Min      Max      Average
  Mem Avail:      7MB    2960MB    2516MB
  Processes:     42      44      42
  Threads:      109     112     109
```

Мною была реализована программа по “поеданию памяти” у устройства, посредством выделения квадратичной матрицы в бесконечном цикле при помощи malloc(), при значении n равном 100400, соответственно 100400^2 при попытке выделить больше случается Memory fault, о чем свидетельствует скриншот ниже.

```
4, b - 100400
100400
Memory fault (core dumped)
```

Выходит что для того чтоб использовать всю память в моем случае, необходимо выделить 2G памяти, что в целом и свидетельствует top.



The screenshot shows a terminal window with two panes. The left pane, titled 'tty0: sem', shows the execution of a program 'sem' which results in a 'Memory fault (core dumped)'. The right pane, titled 'tty1: top', shows the output of the 'top' command, displaying system statistics and a list of running processes.

```
tty0: sem
4, b - 1410065408
1410065408
4, b - 1215752192
1215752192
4, b - -727379968
# g++ -o sem sem.c
# ./sem
100000000
# g++ -o sem sem.c
# ./sem
100000000
# g++ -o sem sem.c
# ./sem
100000
4, b - 100100
100100
4, b - 100200
100200
4, b - 100300
100300
4, b - 100400
100400

tty1: top
44 processes: 112 threads:
CPU states: 1.3% idle, 12.6% user, 85.9% kernel
Memory: 0 total, 7M avail, page size 4K

  PID   TID  PRI  STATE   HH:MM:SS   CPU  COMMAND
    1     6   10   Run     0:00:31  47.08% kernel
    1    14   10   Rcv     0:00:23  38.76% kernel
   8200    9   10   Rcv     0:00:01   5.74% devb-eide
   8200    2   21   Rcv     0:00:03   3.66% devb-eide
  172052    1   10   Rdy     0:00:01   2.56% dumper
 1638438    1   10   Rcv     0:00:08   0.29% firefox-bin
   8200    4   21   Rcv     0:00:00   0.13% devb-eide
    1    13   10   Rcv     0:00:08   0.13% kernel
  237596    3   12   Rply    0:02:41   0.05% io-graphics
  159759    1   10   Rcv     0:00:50   0.03% devc-pty

      Min      Max      Average
CPU idle:      0%      99%      82%
Mem Avail:    7MB  2953MB  2590MB
Processes:    42    44    42
Threads:     109   112   109
```


Используемые ресурсы

1. Документация QNX - [url]
http://www.qnx.com/developers/docs/6.5.0/index.jsp?topic=%2Fcom.qnx.doc.neutrino_user_guide%2Flimits.html
2. Документация C - [url]
<https://devdocs.io/c/>