ĐẠI HỌC CÔNG NGHỆ - ĐHQGHN

**KHOA ĐIỆN TỬ VIỄN THÔNG**

----- ☙ 🕮 ❧ -----


**Thực tập thiết kế hệ thống nhúng**

**Nhóm thực tập Nhúng- Dự án gNode 5G VHT**

**Báo cáo tuần 1**

# LINUX PROGRAMMING ASSIGMENT

Sinh viên thực hiện:

**Đỗ Thái Vũ - 19021540**

**Hà Nội, ngày 15 tháng 7 năm 2022**

## 1. Viết chương trình C trên Linux chạy 3 thread SAMPLE, LOGGING, INPUT. Trong đó:

1.1 Thread SAMPLE

- Thread **SAMPLE** thực hiện vô hạn lần nhiệm vụ sau với chu kì **X** ns. Nhiệm vụ là đọc thời gian hệ thống hiện tại (chính xác đến đơn vị ns) vào biến **T**.

```
void *getTime(void *args )
{
    clock_gettime(CLOCK_REALTIME,&request1);
  while(check_loop == 1)
  {
    clock_gettime(CLOCK_REALTIME,&tp);
    long temp;
      if(request1.tv_nsec + freq > 1000000000)
      {
        temp = request1.tv_nsec;
```

```
  request1.tv_nsec =0;
  request1.tv_sec +=1;
    if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME, &request1,NULL) != 0)
     {
        check_loop = 0;
     }
     else
     {
       request1.tv_nsec+=temp-1000000000+ freq;
        if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME, &request1,NULL) != 0)
        {
            check_loop = 0;
        }
        else
        {
            check_loop = 1;
        }
     }
}
else{
    request1.tv_nsec +=freq;
    if(clock_nanosleep(CLOCK_REALTIME,TIMER_ABSTIME, &request1,NULL) != 0)
     {
        check_loop = 0;
     }
    else
     {
        check_loop = 1;
     }
}
```

```
  }
}
```

Lưu ý :

- Khi tín hiệu được phân phối ở tốc độ cao thì nanosleep có thể dẫn đến sự thiếu chính xác lớn trong cả 1 quy trình (do thời gian sleep), để giải quyết vấn đề, lần call đầu tiên dùng hàm clock_gettime() để truy xuất thời gian, sau đó thêm thời lượng mong muốn vào. Cuối cùng gọi clock_nanosleep() với TIMER_ABSTIME flag.

1.2. Thread INPUT

- Thread **INPUT** kiểm tra file “freq.txt” để xác định chu kỳ **X** (của thread **SAMPLE**) có bị thay đổi không?, nếu có thay đổi thì cập nhật lại chu kỳ X. Người dùng có thể echo giá trị chu kỳ X mong muốn vào file “freq.txt” để thread **INPUT** cập nhật lại **X**.

```
void *getFreq(void *args)
{
  while(1)
  {
  pthread_mutex_lock(&mutex);
  FILE *fp;
  fp = fopen("freq.txt","r");
   unsigned long new_freq = 0;
  fscanf(fp,"%lu",&new_freq);
  fclose(fp);
  if(new_freq == freq)
  {  pthread_mutex_unlock(&mutex);
```

```
  }
    else
  {
    freq = new_freq;
    pthread_mutex_unlock(&mutex);
  }
  }
}
```

## 1.3. Thread LOGGING

- Thread **LOGGING** chờ khi biến **T** được cập nhật mới, thì ghi giá trị biến **T** và giá trị **interval** (offset giữa biến **T** hiện tại và biến **T** của lần ghi trước) ra file có tên “time_and_interval.txt”.

```
void *save_time(void *args)
{
  while(1)
  {
    FILE *file;
    file = fopen("interval.txt","a+");
    long diff_sec = ((long) tp.tv_sec) - tmp.tv_sec ;
    long diff_nsec;
    if(tmp.tv_nsec != tp.tv_nsec || tmp.tv_sec != tp.tv_sec)
    {
      if(tp.tv_nsec > tmp.tv_nsec)
        {
```

```
        diff_nsec = tp.tv_nsec - tmp.tv_nsec;
      }
    else
      {
        diff_nsec = 1000000000 + tp.tv_nsec - tmp.tv_nsec ;
        diff_sec = diff_sec - 1;
      }
     fprintf(file,"\n%ld.%09ld",diff_sec,diff_nsec);
     tmp.tv_nsec =tp.tv_nsec;
     tmp.tv_sec = tp.tv_sec;
  }
  fclose(file);
}
```

## 2. Shell script để thay đổi giá trị chu kỳ X

Trong phần này, ta sẽ viết shell script để thay đổi lại giá trị chu kỳ X trong file “freq.txt” sau mỗi 1 phút. Và các giá trị X lần lượt được ghi như sau: 1000000 ns, 100000 ns, 10000 ns, 1000 ns, 100ns

File script.sh

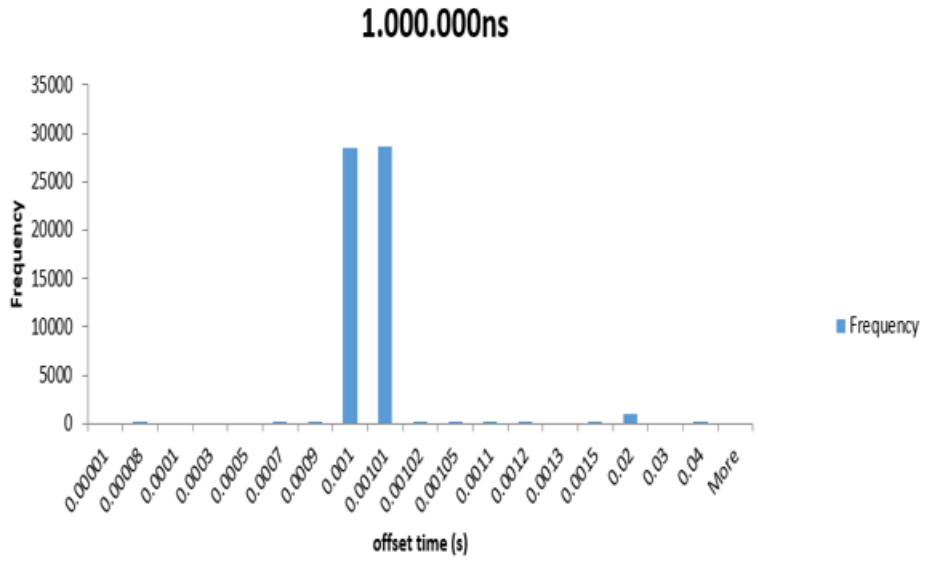
```
#!/bin/sh
echo "1000000">freq.txt
timeout 60s ./b1
echo "100000">freq.txt
timeout 60s ./b1
echo "10000">freq.txt
timeout 60s ./b1
echo "1000">freq.txt
timeout 60s ./b1
echo "100">freq.txt
timeout 60s ./b1
```

## 3. Thực hiện chạy shell script + chương trình C

Trong phần này, ta sẽ cho chạy chương trình C và shell script trong vòng 5 phút, và sau đó dừng chương trình C (theo lệnh trên thì chương trình mỗi freq chỉ chạy trong 60 s).

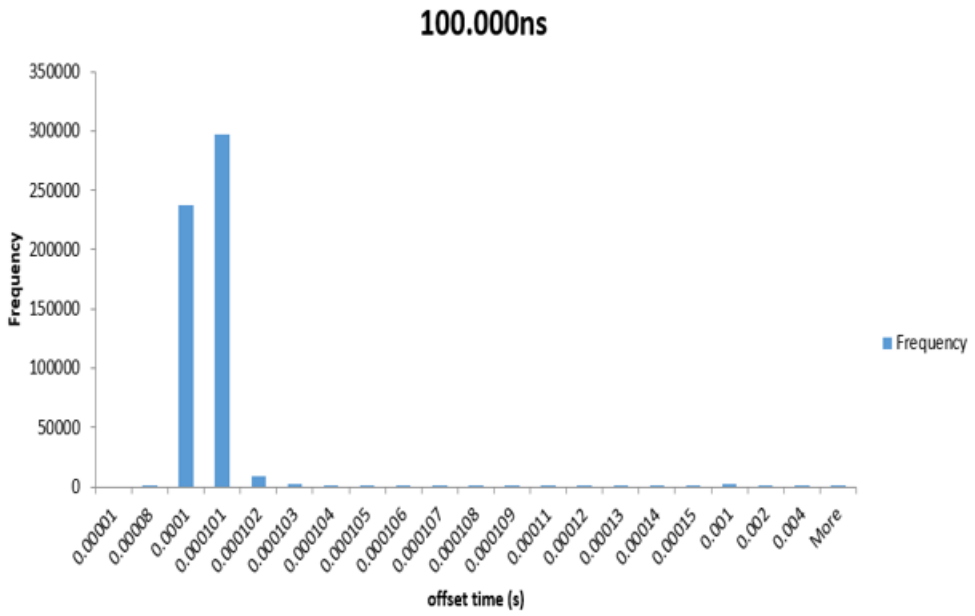
## 4. Khảo sát giá trị interval

4.1. Với chu kỳ X = 1000000ns




Hình 4.1 Biểu đồ histogram khi chu kỳ X = 1000000ns

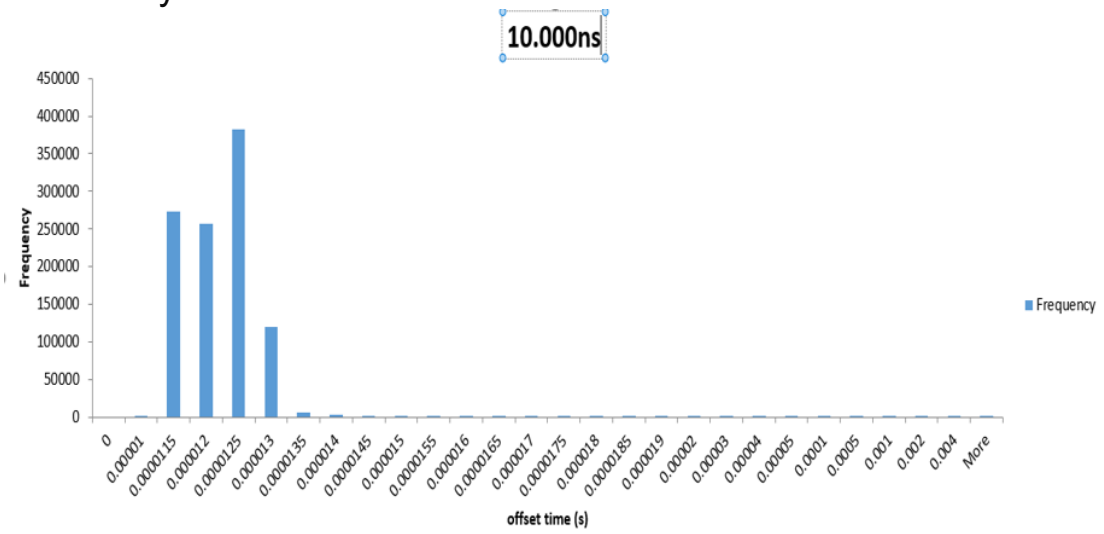
Nhận xét : ta có thể thấy các interval chủ yếu ở 0.001s và 0.00101s, chủ yếu trong khoảng [0.001-0.002].4.2. Với chu kỳ X = 100000ns




Hình 4.2 Biểu đồ histogram khi chu kỳ X = 100000ns

Nhận xét : Các interval xuất hiện nhiều ở offset 0.0001s và 0.000101s , chủ yếu nằm trong khoảng [0.0001-0.00012]

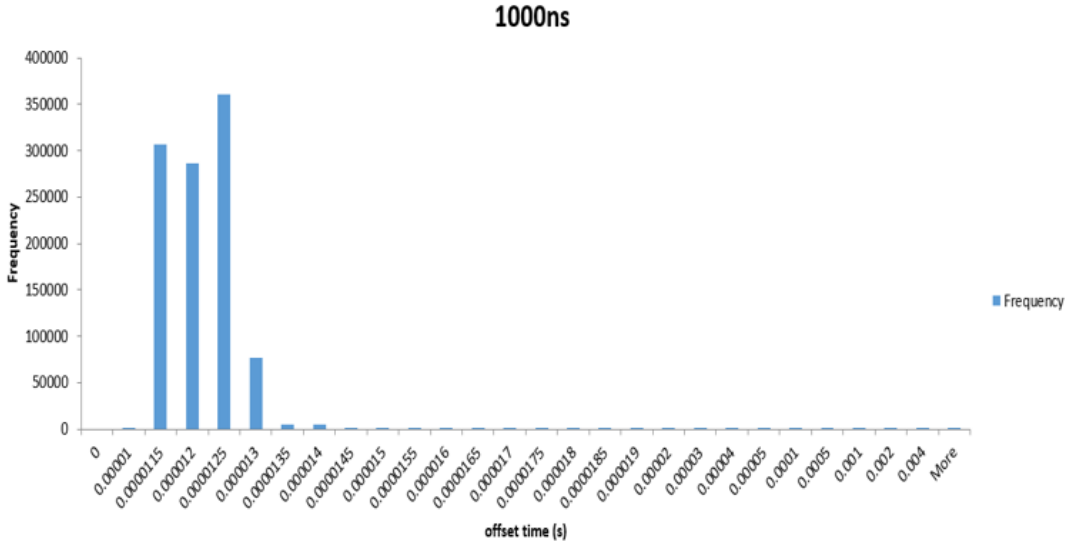
4.3. Với chu kỳ X = 10000ns




Hình 4.3 Biểu đồ histogram khi chu kỳ X = 10000ns

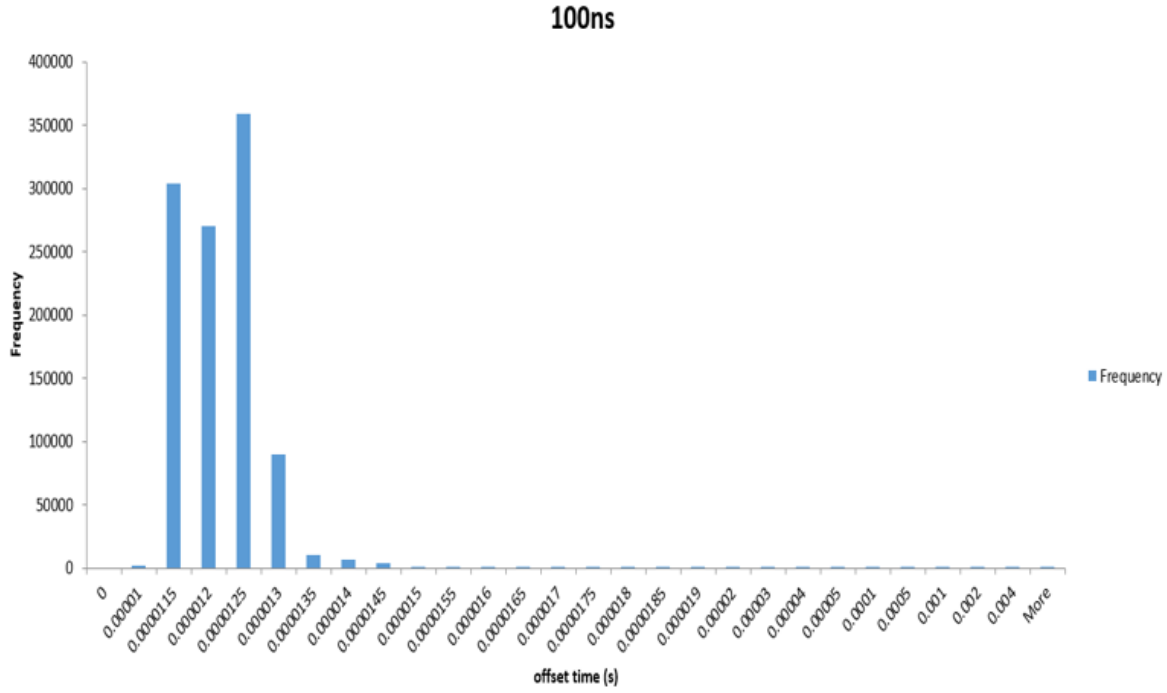
Nhận xét: Các Interval xuất hiện nhiều ở offset 0.0000125s, 0.000012, 0.00001s

4.4. Với chu kỳ X = 1000ns




Hình 4.4 Biểu đồ histogram khi chu kỳ X = 1000ns

4.5. Với chu kỳ X = 100ns




Hình 4.5 Biểu đồ histogram khi chu kỳ X = 100ns

Nhận xét chung :

- Khi chu kỳ càng lớn thì giá trị interval giữa T hiện tại và T của lần ghi trước càng lớn.

- Trong các lần lấy mẫu đều xuất hiện thời gian lớn (0.003s)

## 5. Kết luận

- Tiến độ công việc: Đã hoàn thành trước deadline (15/7/2022)

- Khó khăn gặp phải :

+Shell Script trên Hệ ĐH Linux

+ Kiến thức về C như Threading , Set time

+ Chưa tối ưu được thuật toán một cách tối ưu: Giá trị interval thu được chưa đạt như mong muốn;