

# Homework #3

[ECE30021/ITP30002] Operating Systems

# Mission

---



- Practice multithreaded programming
  - Solve the problems on the server using vi and gcc.
  - Don't use other editor or compiler
- Submission
  - Submit hw3.c on HISNet
- Due date: PM 11:00, Apr. 17<sup>th</sup>

# Honor Code Guidelines



## ■ “Assignment”

- Assignments are an educational activity necessary to fully understand the lecture, and to apply the materials to practical problems. Students should complete all assignments with honesty and sincerity to develop the knowledge and skills intended in the assignment.
- Submitted assignments are reflected in a grade evaluation. Therefore, student should never commit any kinds of dishonest behavior including acquisition, utilization, request, and providing of unauthorized information or assistance that can disrupt the fairness of evaluation.
- It is allowed to get help from peers in order to understand the lecture materials, announcements, or the directions for homework. However, each student should complete the entire assignments on their own.
- Submitting assignments or program codes written by others or acquired from the internet without explicit approval of the professor is regarded as cheating.
- Showing or lending one's own homework to other student is also considered cheating that disturbs fair evaluation and hinders the academic achievement of the other student.
- It is regarded as cheating if two or more students conduct their homework together and submit it individually when the homework is not a group assignment.
- One may receive help from other students if a problem in the assignment could not be solved even with the best effort; however, the assistance should never go beyond the correction of basic grammatical errors. It is cheating to get help about design, logic, and algorithm required in assignments from other person than the professor, TA, and tutor.
- It can be suspected or regarded as cheating if the similarity between assignments submitted by different students is beyond an acceptable degree that can be considered as a coincidence or when the student is not able to explain in detail about their homework.

# Honor Code Guidelines

## ■ “과제”

- 과제는 교과과정의 내용을 소화하여 실질적인 활용 능력을 갖추기 위한 교육활동이다. 학생은 모든 과제를 정직하고 성실하게 수행함으로써 과제에 의도된 지식과 기술을 얻기 위해 최선을 다해야 한다.
- 제출된 과제물은 성적 평가에 반영되므로 공식적으로 허용되지 않은 자료나 도움을 획득, 활용, 요구, 제공하는 것을 포함하여 평가의 공정성에 영향을 미치는 모든 형태의 부정행위는 단호히 거부해야 한다.
- 수업 내용, 공지된 지식 및 정보, 또는 과제의 요구를 이해하기 위하여 동료의 도움을 받는 것은 부정행위에 포함되지 않는다. 그러나, 과제를 해결하기 위한 모든 과정은 반드시 스스로의 힘으로 수행해야 한다.
- 담당교수가 명시적으로 허락한 경우를 제외하고 다른 사람이 작성하였거나 인터넷 등에서 획득한 과제물, 또는 프로그램 코드의 일부, 또는 전체를 이용하는 것은 부정행위에 해당한다.
- 자신의 과제물을 타인에게 보여주거나 빌려주는 것은 공정한 평가를 방해하고, 해당 학생의 학업 성취를 저해하는 부정행위에 해당한다.
- 팀 과제가 아닌 경우 두 명 이상이 함께 과제를 수행하여 이를 개별적으로 제출하는 것은 부정행위에 해당한다.
- 스스로 많은 노력을 한 후에도 버그나 문제점을 파악하지 못하여 동료의 도움을 받는 경우도 단순한 문법적 오류에 그쳐야 한다. 과제가 요구하는 design, logic, algorithm의 작성에 있어서 담당교수, TA, tutor 이외에 다른 사람의 도움을 받는 것은 부정행위에 해당한다.
- 서로 다른 학생이 제출한 제출물간 유사도가 통상적으로 발생할 수 있는 정도를 크게 넘어서는 경우, 또는 자신이 제출한 과제물에 대하여 구체적인 설명을 하지 못하는 경우에는 부정행위로 의심받거나 판정될 수 있다.

# Problem 0

---



- Search Internet for the following functions and read them carefully.
  - `pthread_attr_init()` – initialize `pthread_attr_t` variable
  - `pthread_create()` – create a thread
  - `pthread_join()` – wait for a thread
  - `pthread_exit()` – terminate a thread
- Read the following document to understand variable argument in C.
  - <https://www.eskimo.com/~scs/cclass/int/sx11b.html>

# Problem 0

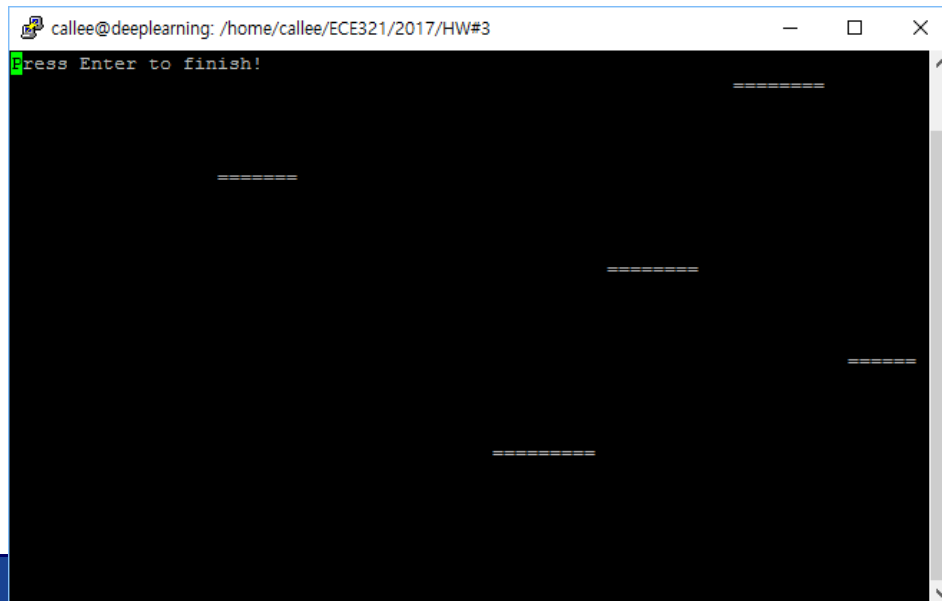
---



- Read the following document to understand variable argument in C.
  - <https://www.eskimo.com/~scs/cclass/int/sx11b.html>
- Read hw3\_skeleton.c carefully to understand all techniques in it.
  - Don't have to understand gotoxy() and clrscr()

# Goal

- Write a multi-threaded program that animates bouncing horizontal bars.
  - Read <# of bars> from command line arguments.
  - The bars are different in length and speed.
  - Each bar is moved by a separate thread.
    - 5 bars requires 5 threads



The screenshot shows a terminal window with the title bar "callee@deeplearning: /home/callee/ECE321/2017/HW#3". The terminal content includes the prompt "Press Enter to finish!" followed by five horizontal bars of varying lengths and positions, representing the animation of bouncing bars. The bars are represented by strings of equals signs (=) on a black background.

# Example

```
$ gcc hw3.c -o hw3 -lpthread
```

```
$ ./hw3 5
```

// bounces 5 bars



```
callee@deeplearning: /home/callee/ECE321/2017/HW#3
Press Enter to finish!

=====
=====
=====
=====
=====
```

The terminal window displays a bar chart with 5 bars of increasing height. The first bar is the shortest, and each subsequent bar is taller than the previous one. The bars are represented by horizontal lines of equal length, with the height of each bar corresponding to its position in the sequence. The terminal window title is 'callee@deeplearning: /home/callee/ECE321/2017/HW#3' and it has standard window controls (minimize, maximize, close). A green cursor is visible at the prompt 'Press Enter to finish!'.



# 1. Structure and Global Variables

- Define a structure *BarInfo* for the argument of the thread function.
- The structure should contain the following fields
  - `int x, y;` // the coordinate of left endpoint
  - `int dx;` // direction (-1: left, +1: right)
  - `int len;` // the length of bar
  - `int interval;` // update interval to control speed
- Declare global variables
  - `int gThreadContinue = 1;`
  - `pthread_mutex_t mutex;`

## 2. Write Thread Function

- Write a function “void\* MoveBarThreadFn(void \*param)” to animate a bouncing bar.
  - Cast *param* to “*BarInfo\**” type  
Ex) `BarInfo *barInfo = (BarInfo *) param;`
  - Repeat until `gThreadContinue` becomes zero.
    - Erase bar at previous coordinate
      - Call `DrawBar()` function with information in *barInfo*.
    - Check and handle bouncing
      - If  $(x + dx)$  is smaller than 1 or  $(x + dx + len)$  is larger than screen width, change the sign of  $dx$ ;
    - Update bar coordinate by adding  $dx$  to  $x$ .
    - Draw bar at new coordinate
    - Sleep for *interval* microseconds.

### 3. Write main Function



- Write main() to retrieve <# of threads> from command line parameter launching threads.
  1. Retrieve *noThread* (# of threads) from the command line arguments. (use `atoi(argv[1])`)
  2. Dynamically allocate an array of *pthread\_t* type (name it *tid*)
  3. Dynamically allocate an array of *BarInfo* type (name it *barInfo*)
    - The length of *tid* and *barInfo* should be <# of threads>.
    - On memory allocation failure, print appropriate message.
    - Initialize `barInfo[t]`

### 3. Write main Function (cont'd)

- Write main() to retrieve <# of threads> from command line parameter launching threads.
  - Initialize barInfo[t] for all  $t$  in  $[0, \text{<\# of threads>})$ 
    - $x = 1$
    - $y = (\text{SCREEN\_HEIGHT} - 1) / \text{<\# of threads>} * t + 2$
    - $dx = 1$  // right direction
    - len = a random number between 5 and 10
    - Interval = (a random number between 1 and 100) \* 1000
  - Create threads to run MoveBarThreadFn() passing &barInfo[t]'s as arguments
  - Wait for Enter key by calling getchar().
  - Terminate threads by assigning zero to *gThreadContinue*.
  - Wait for all children.
  - Call clrscr() and print a good bye message.
  - Deallocate memory