**UNIX Programming Assignment 2**

**Due date : 2016/10/4 23:59**

**Demo time: 2016/10/4 18:30~22:00**

**Demo room: EECS 328**

In this assignment, you will need to write your own dup2 function that behaves the same way as the dup2 function described in Section 3.12.

**int dup2(int** *oldfd***, int** *newfd***);**

The **dup2**() system call perform a copy of the file descriptor *oldfd,* using the newfd as target file descriptor.

* Your dup2 can copy the file descriptor *oldfd* and use the *newfd* as target.
* Make sure you copy the *oldfd* to the *newfd* correctly.
* The return file descriptor should be the new *newfd* which point to the file table of *oldfd*. If error occur, you have to return -1.
* If *newfd* is not close, you have to close *newfd* before you perform copy.
* Your dup2 can handle invalid file descriptor (You may want to check which value satisfy the valid file descriptor online) and others error status.
* Note that you can not use **dup2, fcntl functions** in your implementation.

You need to implement your dup2 function (named **mydup2**) in your .c file(s) and use header file .h (**named hw2.h**) to declare your implementation while demo.

Hint: If **fcntl** cannot be invoked, you will have to use **dup**. Then you have no control over which file descriptor will be used by the dup function call. Try to design a workaround of this.

Hint: You may want to handle the maximum file descriptors (OPEN\_MAX) which a single process can use when you design your dup2. If OPEN\_MAX is not defined, you will have to find the maximum file descriptors. (You can use the open\_max() source code given by TA.)

Submission:

* You have to upload the assignment to the ilms system.
* You also need to **demo and explain your implementation to the TA**.
* (1%) Submit you pseudocode in plan-text, with the file name: hw02\_[YourStudentID].txt
* (3%) Submit your code with the file name: hw02\_[YourStudentID].c and the header file (hw2.h). You get 3 points once your code can handle normal test cases prepared by TA.
  + Your dup2 can duplicate the descriptor correctly.
  + If newfd is a open file descriptor, you have to close the newfd first.
  + if oldfd is equal to newfd, your dup2 will return newfd.
* (1%) You get one more point if you handle all the errors correctly.
  + If oldfd is a invalid file descriptor, your dup2 will return -1. Newfd will remain its original status.