**ASSIGNMENT – 2**

1. **Objective:**

Our objective in this assignment 3 is to create Nucleus message passing using nachos and this is done with the help of four system calls. These four system calls are SendMessage, WaitMessage, SendAnswer, WaitAnswer. In this assignment, nucleus here refers to kernel in nachos.

1. **What has been developed:**

**SendMessage SYS\_CALL:**

In this system call, at first we take the three parameters which are receiver, message and bufferId from the function called in .c file and save them. We check if bufferId is equal to -1 and if it has then we directly call HandleSendMessage() function else we take an Id from bitmap and then send it to HandleSendMessage() and if no available bitmap present then we return -1 using WriteRegister. When we call this function, first we convert char\* of receiver and sender to string and create a message object which stores sender, receiver and message. We then check if threadMap(stores key as string and and value as thread) of receiver is not null, if yes then get the buffer from the list of buffers present in thread, else we clear bitmap and return -1 as there is no process existed. If buffer is null, we have to append this buffer to the bufferpool list of the sender and receiver threads and append the message to buffer. Subsequently, we check if the thread status is blocked, If yes then wake up the thread by ReadyToRun() and then return bufferId using the WriteRegister().

**WaitMessage SYS\_CALL:**

In this system call, at first we take the three parameters which are receiver, message and bufferId from the function called in .c file and save them like previous system call. Then we call the HandleWaitMessage() where it handles the messages that are sent from other process. Here we take the current thread running which handles this message. We then convert char\* sender to string data type sender and get the bufferId from current thread buffermap. Then we check if the bufferId what we got is present in current thread bufferpool. If yes then assign that buffer to a variable. If buffer is not null then break else sleep the thread. Later, we check if the current thread buffer does not have message for it to send then sleep the thread again and then come out of sleep if it gets and print the message what we got and return the bufferId.

**SendAnswer SYS\_CALL:**

In this system call, we get the same parameters like the before system calls and store them and send them as parameters to SendAnswer function. At first we collect the answer from the second parameter and get it through ReadMem function. We start sending answer now, as we check if the bufferId what we got from third parameter is present in current threads bufferpool, if yes then store in a variable. If buffer is not null, we create a message object and store the data of sender, isAnswer(TRUE) , answerType , receiver and message and append the message to the buffer. If threadMap thread of receiver is not null and status is blocked then we wake up the thread using ReadyToRun() else we say that the receiver is finished. If buffer is null then buffer is not present and we close the sendAnswer function which sends the message to the sender.

**WaitAnswer SYS\_CALL:**

In this system call, we get the same parameters like the before system calls and store them and send them as parameters to WaitAnswer function. Here we first check if bufferId is -1, if yes we say that the message will be failed and return to the next ongoing thread. Then we check if the buffer is existing in the current thread bufferpool, if yes then store the buffer and come out of the loop. Here, we check if buffer is not null, if yes then come out of loop else set thread status to blocked and sleep the thread. If it is activated i.e, the thread is woke up, then we ch

1. **How to test your solution:**

Unzip the file and go to path nachos/code/build.linux. Since makefile dependencies are cleared, no need to use make clean command. Then use the following 3 commands to get the output:

* 1. make depend
  2. make
  3. ./nachos -x ../test/[FileName1] x ../test/[FileName2] x ../test/[FileName3] and so on until the umber of files you want to run.

**(OR)**

./nachos -quantum [Number] -x ../test/[FileName1] -x ../test/[FileName2]

-x ../test/[FileName3] and so on until the umber of files you want to run.

1. **Files modified / added:**

The files modified are main.cc, addrspace.cc, addrspace.h, syscall.h, exception.cc in the userprog directory and kernel.cc, kernel.h, thread.cc, thread.h in threads directory and makefile in test directory and makefile in build.linux.

The files added are process1, process2, process3, process4 and process5 in test directory and buffer.cc, buffer.h, message.cc, message.h.

1. **Output:**
2. **Signed Disclosure Form:**

Assignment #: 3

Name: Rumalla Karthik , Adway Malhotra

1. Did you consult with anyone other than instructor or TA/grader on parts of this assignment?

If Yes, please give the details.

Ans 1) Yes, I have asked some doubts to TA in the Lab sessions.

2. Did you consult an outside source such as an Internet forum or a book on parts of this assignment?

If Yes, please give the details.

Ans 2) Yes, I have read an IEEE paper named “The nucleus of multiprogramming system” for this assignment.

I assert that, to the best of my knowledge, the information on this sheet is true.

Signature: \_\_\_\_ Rumalla Karthik \_ ,\_ Adway Malhotra \_\_\_\_ Date : 12/7/2018