# Nachos: Multiprogramming–Resolving the First Limitation

## 1 An Important Reminder

Review the university's academic integrity policy (http://class.syr.edu/academic-integrity). Remember that violating academic integrity policy will significantly jeopardize your grade.

## 2 Learning Objectives

After completing this project, students can

1. modify the main program so it can read multiple user program names,

## 3 Preliminary Requirement

- Get a fresh copy of Nachos from `nachos.tar`. Refer to TA's lecture.
- Download Project 1 files from Bb and place them in the right directories in your Nachos directory. These are `main.cc` and `test1.tar`.
- Compile Nachos with the new `main.cc`. Unarchive `test1.tar` and compile all user programs in that directory.

## 4 Overview

The original distribution of Nachos implements uni-programming, allowing one program execution at a time. In TA's lecture, we studied three limitations in Nachos in supporting multiprogramming. In this project, you need to resolve the first limitation. To focus on fixing the first limitation, I provide a modified `main.cc`. You need to modify this file further to complete the project.

Recall that the distribution version of Nachos runs a user-level program from the `main` thread. See `RunUserProg ()` in `main.cc`. The following command runs the user program `write`:

```
./nachos -x ../test1/write
```

However, this main program will not run a user program; instead, it will print out the user program name and stall. Therefore, you will get the following outputs.

```
build.linux>./nachos -x ../test1/halt
The user program name is ../test1/halt

build.linux>./nachos -x ../test1/write -x ../test1/read
The user program name is ../test1/read

build.linux>./nachos -x ../test1/write -x ../test1/read -x ../test1/halt
The user program name is ../test1/halt
```

## 5 Requirements

You will fix the first limitation in this project. We will discuss the second and third limitation in Project 2. Refer to TA's lecture if needed.

1. By default, Nachos can read only one user program name through the `-x` flag, no matter how many you specify with multiple `-x` flags. Modify `main.cc` so Nachos can take as many user programs via multiple `-x` flags.

## 6 Tests and Output

Your program will be tested at least with the following commands for the correct outputs:

```
build.linux>./nachos -x ../test1/halt
Program [0] = ../test1/halt

build.linux>./nachos -x ../test1/write -x ../test1/read
Program [0] = ../test1/write
Program [1] = ../test1/read

build.linux>./nachos -x ../test1/write -x ../test1/read -x ../test1/halt
Program [0] = ../test1/write
Program [1] = ../test1/read
Program [2] = ../test1/halt
```

## 7 What to Submit

You must submit two files to Bb: (1) The PDF report and (2) The source code zip file.

1. **PDF report:** You need to submit a detailed project report describing what you have done to implement the requirements. Please also list the important code snippets followed by an explanation. These code snippets should include the modified part of the Nachos source code. Simply attaching code without any description will not receive credits.
2. **Source code zip file:** After "`make clean`" in `build.linux`, please compress your whole `nachos` folder, name the compressed file LastName,FirstName.zip (NOTE: .zip ONLY!)