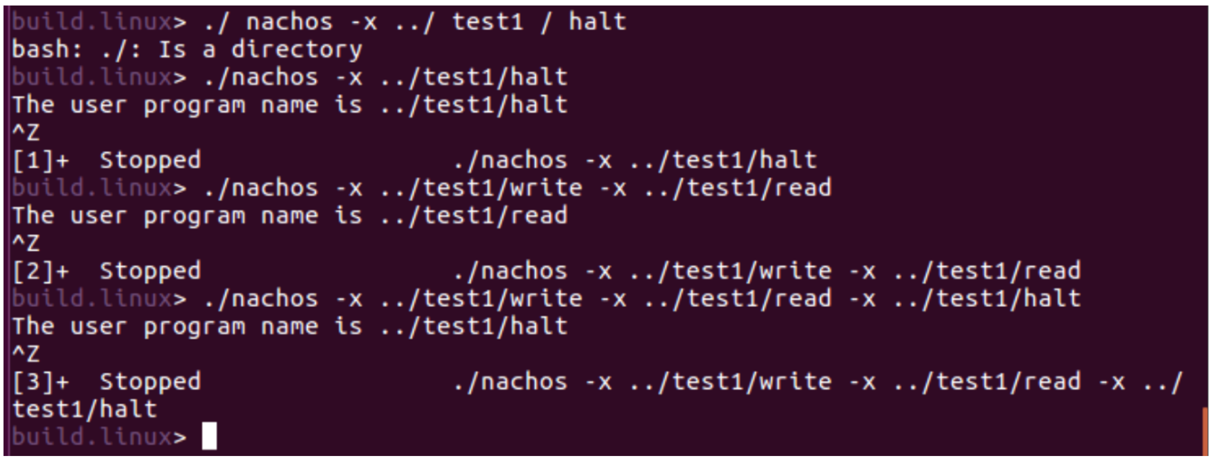
1.The original result of nachos running.



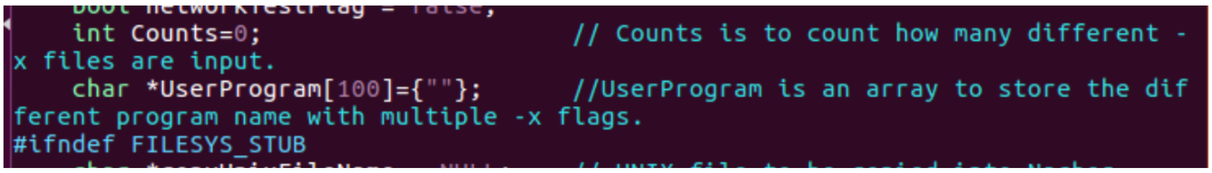
```
build.linux> ./ nachos -x ../ test1 / halt
bash: ./: Is a directory
build.linux> ./nachos -x ../test1/halt
The user program name is ../test1/halt
^Z
[1]+  Stopped                 ./nachos -x ../test1/halt
build.linux> ./nachos -x ../test1/write -x ../test1/read
The user program name is ../test1/read
^Z
[2]+  Stopped                 ./nachos -x ../test1/write -x ../test1/read
build.linux> ./nachos -x ../test1/write -x ../test1/read -x ../test1/halt
The user program name is ../test1/halt
^Z
[3]+  Stopped                 ./nachos -x ../test1/write -x ../test1/read -x ../
test1/halt
build.linux> 
```

The original Outputs

It can only store one newly program name and output for the result in the original code, and stall. Therefore, we will get the outputs

2.Modified code.

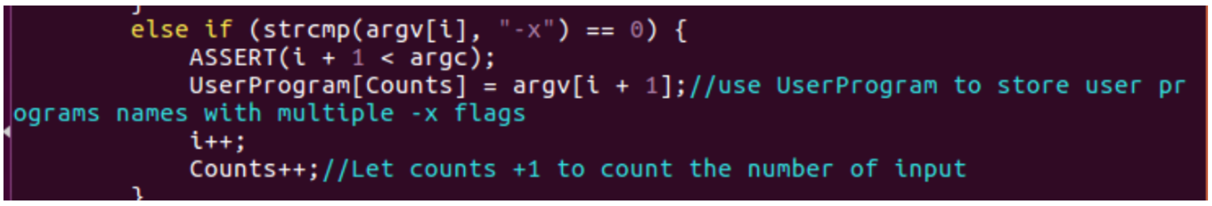
Firstly, we need to add two variables, Counts and UserProgams[]. And its definitions are as follows.



```
    bool networkTestFlag = false;
    int Counts=0;                  // Counts is to count how many different -
x files are input.
    char *UserProgram[100]={""};      //UserProgram is an array to store the dif
ferent program name with multiple -x flags.
#ifndef FILESYS_STUB
```

One to store the number of programs names. And UserProgram to store its locations to prepare for the output.

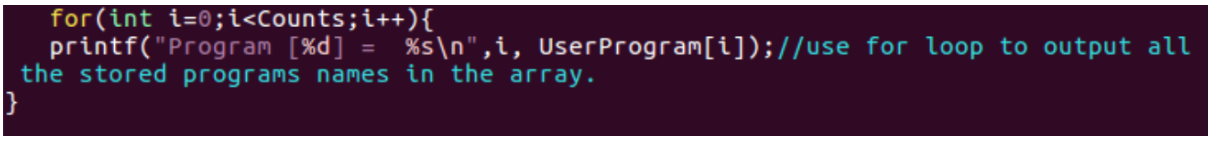
Then we slightly change the function in else if (strcmp(argv[i], "-x") == 0); as follows.



```
        }
        else if (strcmp(argv[i], "-x") == 0) {
            ASSERT(i + 1 < argc);
            UserProgram[Counts] = argv[i + 1];//use UserProgram to store user pr
ograms names with multiple -x flags
            i++;
            Counts++;//Let counts +1 to count the number of input
        }
```

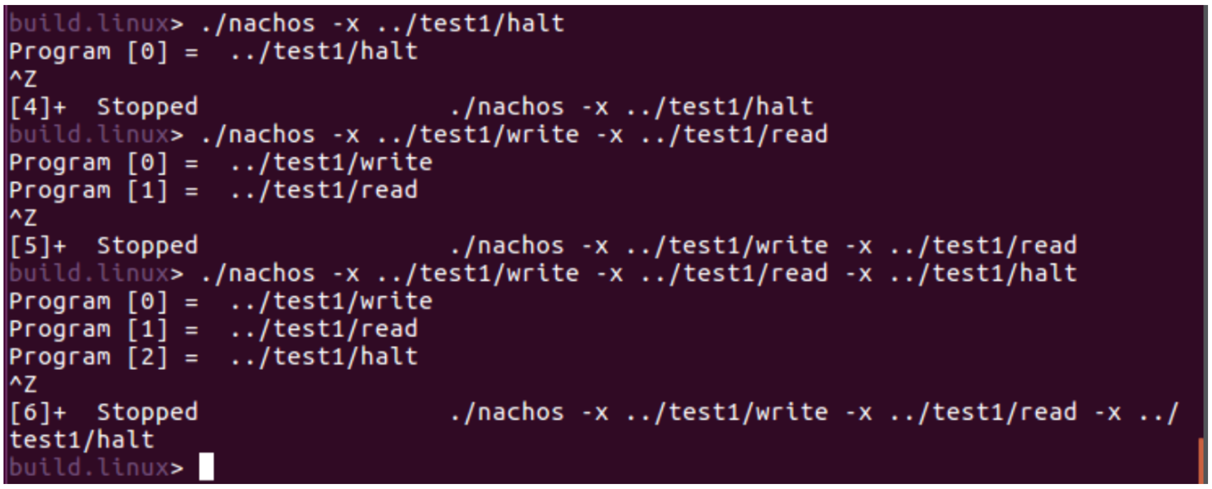
Counts can represent the number of program names, and UserProgram can store the correct program names through multiple -x flags.

Finally, we just need to add a for-loop to output all the stored programs names which are as follows.



```
   for(int i=0;i<Counts;i++){
   printf("Program [%d] =  %s\n",i, UserProgram[i]);//use for loop to output all
 the stored programs names in the array.
}
```

So the output is as follows, which is the same as the requirement in the 6.Test and Outputs



```
build.linux> ./nachos -x ../test1/halt
Program [0] =  ../test1/halt
^Z
[4]+  Stopped                 ./nachos -x ../test1/halt
build.linux> ./nachos -x ../test1/write -x ../test1/read
Program [0] =  ../test1/write
Program [1] =  ../test1/read
^Z
[5]+  Stopped                 ./nachos -x ../test1/write -x ../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
^Z
[6]+  Stopped                 ./nachos -x ../test1/write -x ../test1/read -x ../
test1/halt
build.linux> 
```