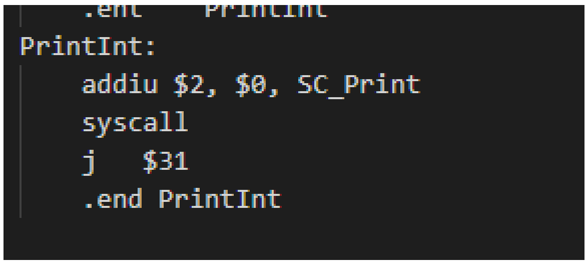
FSS HW01 111598034
郭駿頤

Part 1：Your implementation.

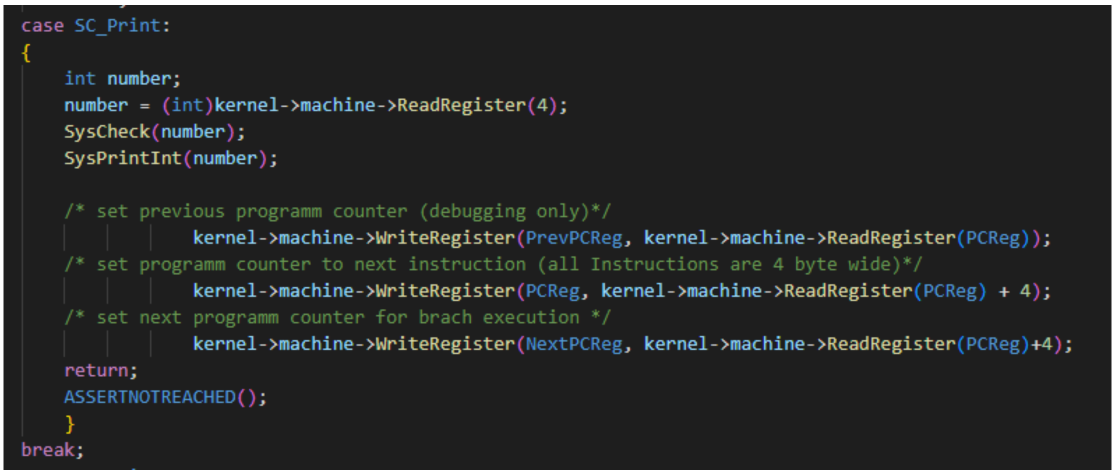
1. PrintInt Systemcall Implement

Step 1. 修改/test/start.S



```
    .ent    PrintInt
PrintInt:
    addiu $2, $0, SC_Print
    syscall
    j   $31
    .end PrintInt
```

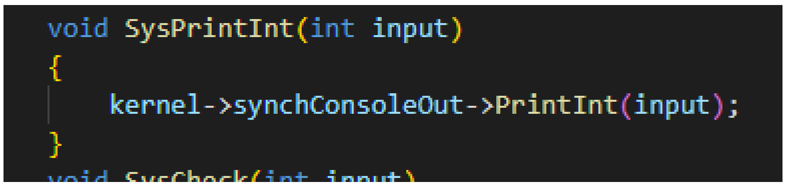
Step 2. 修改/usrprog/exception.cc



```
case SC_Print:
{
    int number;
    number = (int)kernel->machine->ReadRegister(4);
    SysCheck(number);
    SysPrintInt(number);

    /* set previous programm counter (debugging only)*/
            kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
    /* set programm counter to next instruction (all Instructions are 4 byte wide)*/
            kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
    /* set next programm counter for brach execution */
            kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
    return;
    ASSERTNOTREACHED();
    }
break;
```

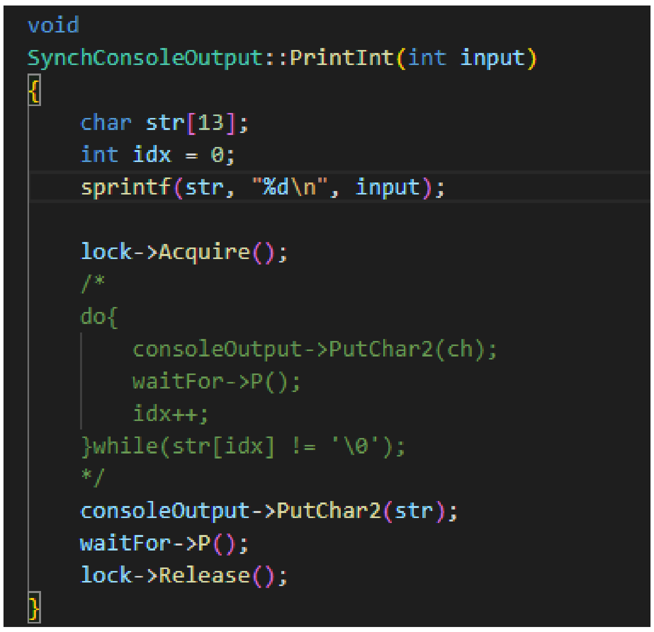
call ksyscall.h SysPrintInt(number)

Step 3. 修改/usrprog/ksyscall.h



```
void SysPrintInt(int input)
{
    kernel->synchConsoleOut->PrintInt(input);
}
void SysCheck(int input)
```

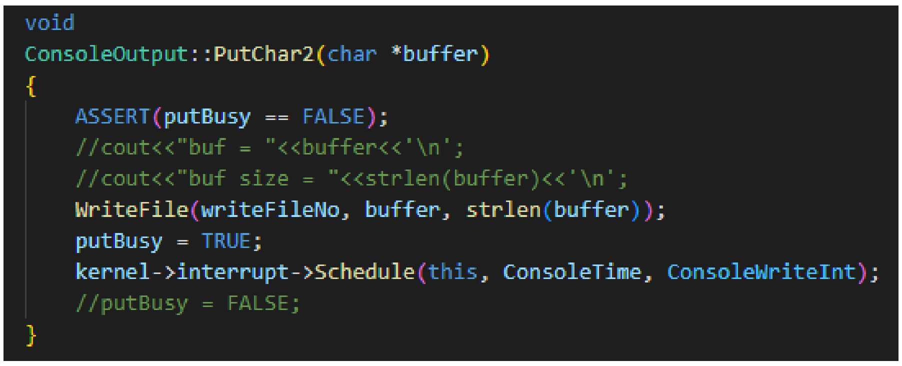
Step 4. 修改/usrprog/synchconsole.cc



```
void
SynchConsoleOutput::PrintInt(int input)
{
    char str[13];
    int idx = 0;
    sprintf(str, "%d\n", input);

    lock->Acquire();
    /*
    do{
        consoleOutput->PutChar2(ch);
        waitFor->P();
        idx++;
    }while(str[idx] != '\0');
    */
    consoleOutput->PutChar2(str);
    waitFor->P();
    lock->Release();
}
```

由於要求需要 writes 4，所以 input 的數字要和\n 一起輸出。

Step 5. 修改/machine/console.cc

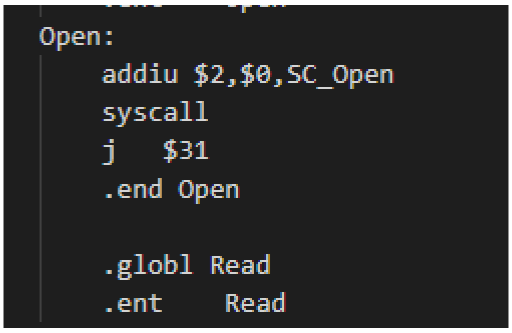


```
void
ConsoleOutput::PutChar2(char *buffer)
{
    ASSERT(putBusy == FALSE);
    //cout<<"buf = "<<buffer<<'\n';
    //cout<<"buf size = "<<strlen(buffer)<<'\n';
    WriteFile(writeFileNo, buffer, strlen(buffer));
    putBusy = TRUE;
    kernel->interrupt->Schedule(this, ConsoleTime, ConsoleWriteInt);
    //putBusy = FALSE;
}
```

負責將輸出到終端。

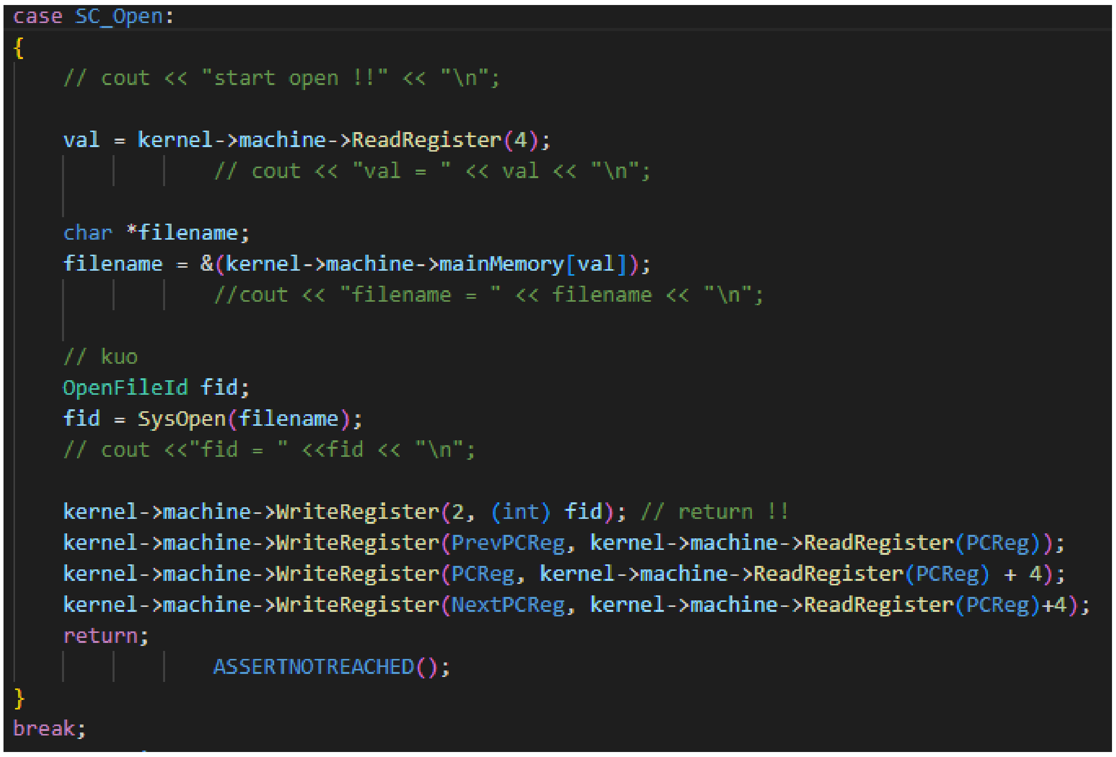
2. Open Systemcall Implement

Step 1. 修改/test/start.S



```
Open:
    addiu $2,$0,SC_Open
    syscall
    j    $31
    .end Open

    .globl Read
    .ent    Read
```

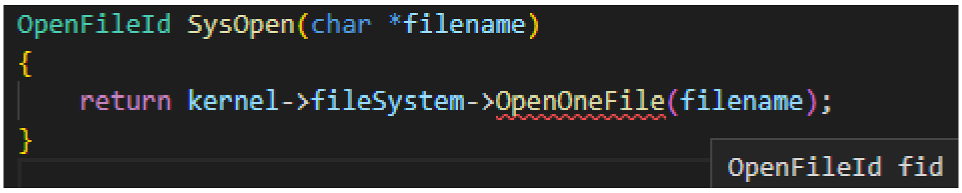
Step 2. 修改/usrprog/exception.cc



```
case SC_Open:
{
    // cout << "start open !!" << "\n";

    val = kernel->machine->ReadRegister(4);
            // cout << "val = " << val << "\n";

    char *filename;
    filename = &(kernel->machine->mainMemory[val]);
            //cout << "filename = " << filename << "\n";

    // kuo
    OpenFileId fid;
    fid = SysOpen(filename);
    // cout <<"fid = " <<fid << "\n";

    kernel->machine->WriteRegister(2, (int) fid); // return !!
    kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
    kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
    kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
    return;
            ASSERTNOTREACHED();
}
break;
```

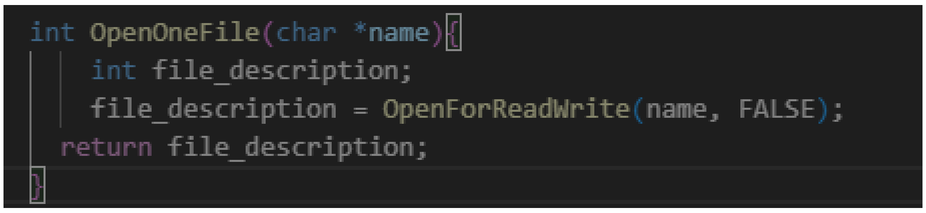
call ksyscall.h SysOpen(number)

Step 3. 修改/usrprog/ksyscall.h



```
OpenFileId SysOpen(char *filename)
{
    return kernel->fileSystem->OpenOneFile(filename);
}
                                        OpenFileId fid
```

Step 4. 修改/filesys/filesys.h

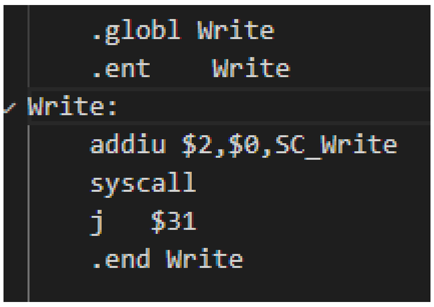


```
int OpenOneFile(char *name){
    int file_description;
    file_description = OpenForReadWrite(name, FALSE);
  return file_description;
}
```

取得 fid。

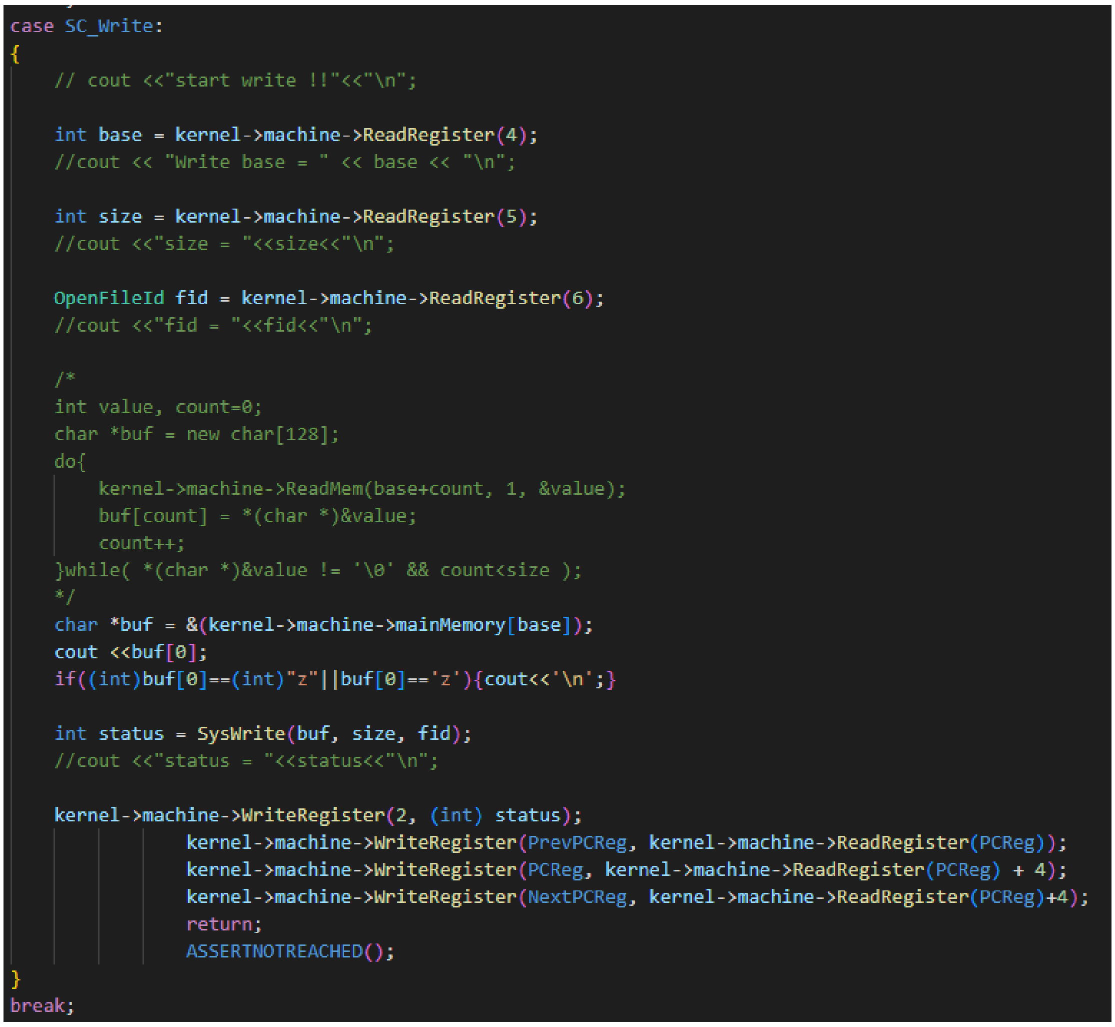
3. Write Systemcall Implement

Step 1. 修改/test/start.S



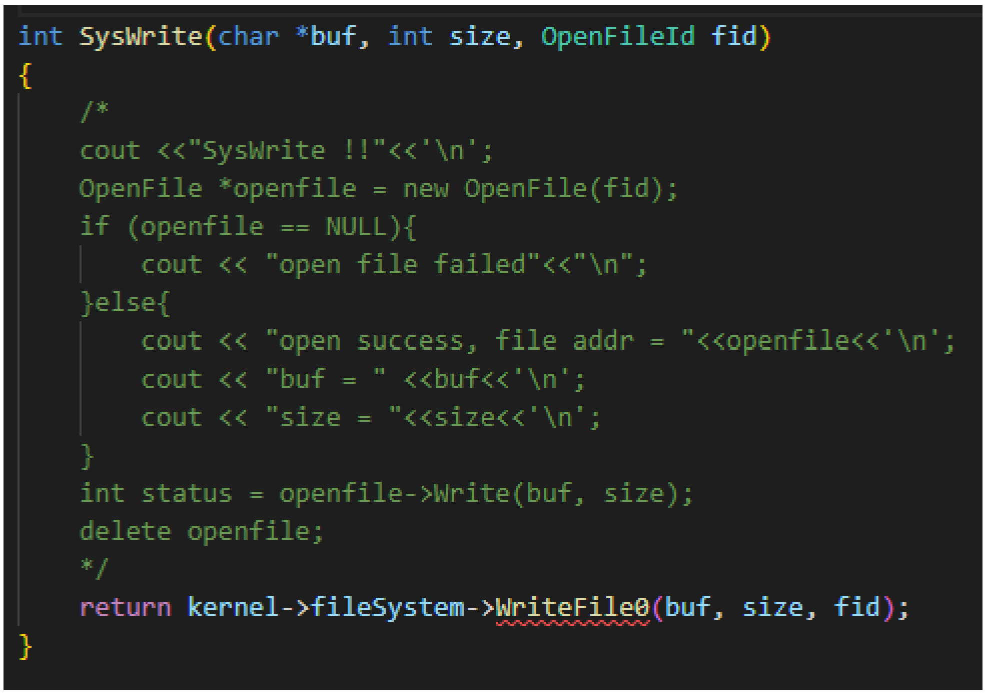
```
    .globl Write
    .ent    Write
Write:
    addiu $2,$0,SC_Write
    syscall
    j   $31
    .end Write
```

Step 2. 修改/usrprog/exception.cc



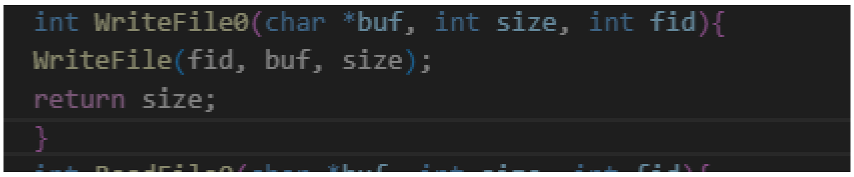
```
case SC_Write:
{
    // cout <<"start write !!"<<"\n";

    int base = kernel->machine->ReadRegister(4);
    //cout << "Write base = " << base << "\n";

    int size = kernel->machine->ReadRegister(5);
    //cout <<"size = "<<size<<"\n";

    OpenFileId fid = kernel->machine->ReadRegister(6);
    //cout <<"fid = "<<fid<<"\n";

    /*
    int value, count=0;
    char *buf = new char[128];
    do{
        kernel->machine->ReadMem(base+count, 1, &value);
        buf[count] = *(char *)&value;
        count++;
    }while( *(char *)&value != '\0' && count<size );
    */
    char *buf = &(kernel->machine->mainMemory[base]);
    cout <<buf[0];
    if((int)buf[0]==(int)"z"||buf[0]=='z'){cout<<'\n';}

    int status = SysWrite(buf, size, fid);
    //cout <<"status = "<<status<<"\n";

    kernel->machine->WriteRegister(2, (int) status);
                kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
                kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
                kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
                return;
                ASSERTNOTREACHED();
}
break;
```

Step 3. 修改/usrprog/ksyscall.h



```
int SysWrite(char *buf, int size, OpenFileId fid)
{
    /*
    cout <<"SysWrite !!"<<'\n';
    OpenFile *openfile = new OpenFile(fid);
    if (openfile == NULL){
        cout << "open file failed"<<"\n";
    }else{
        cout << "open success, file addr = "<<openfile<<'\n';
        cout << "buf = " <<buf<<'\n';
        cout << "size = "<<size<<'\n';
    }
    int status = openfile->Write(buf, size);
    delete openfile;
    */
    return kernel->fileSystem->WriteFile0(buf, size, fid);
}
```

Step 4. 修改/filesys/filesys.h

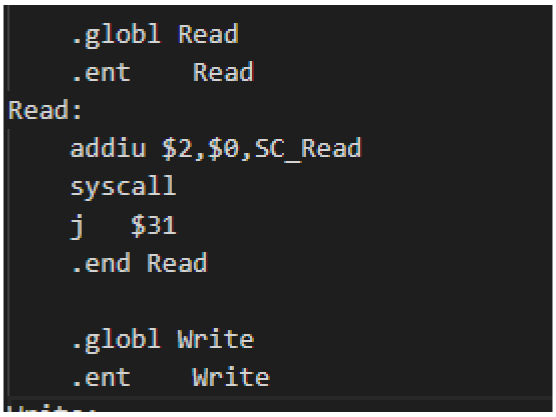


```
int WriteFile0(char *buf, int size, int fid){
WriteFile(fid, buf, size);
return size;
}
```

Return Size .

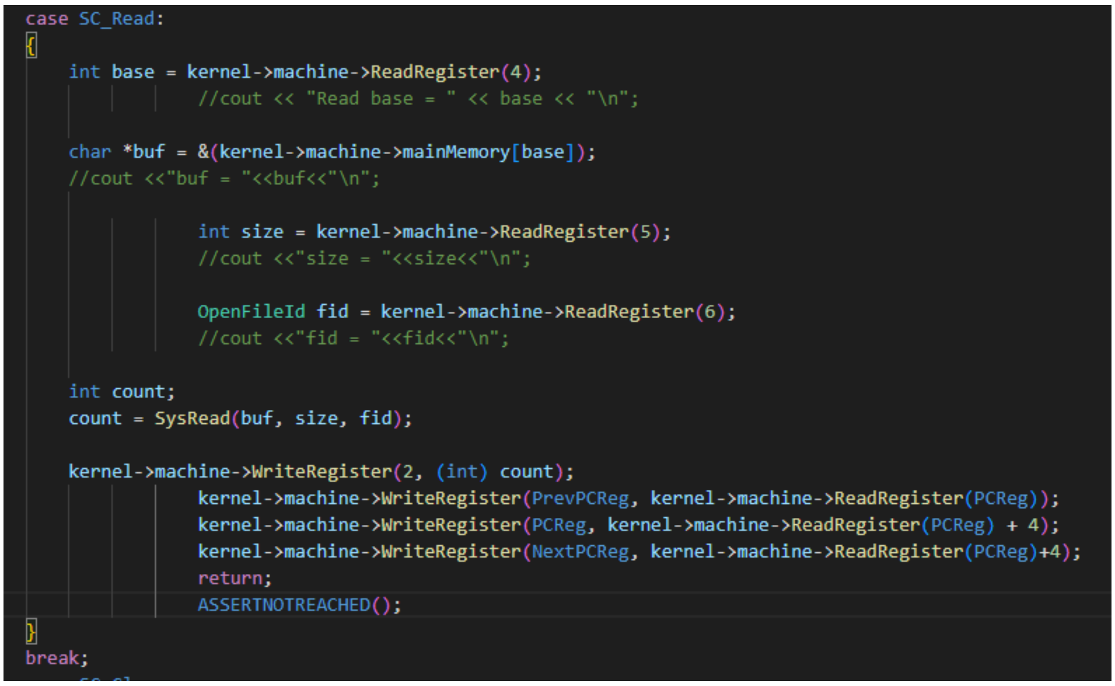
4. Read Systemcall Implement

Step 1. 修改/test/start.S



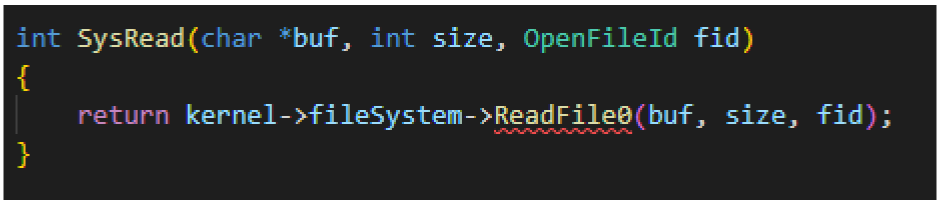
```
    .globl Read
    .ent    Read
Read:
    addiu $2,$0,SC_Read
    syscall
    j   $31
    .end Read

    .globl Write
    .ent    Write
```

Step 2. 修改/usrprog/exception.cc



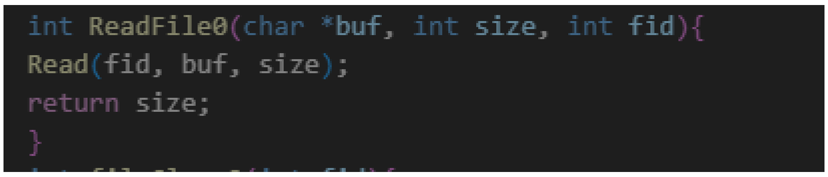
```
case SC_Read:
{
    int base = kernel->machine->ReadRegister(4);
            //cout << "Read base = " << base << "\n";

    char *buf = &(kernel->machine->mainMemory[base]);
    //cout <<"buf = "<<buf<<"\n";

            int size = kernel->machine->ReadRegister(5);
            //cout <<"size = "<<size<<"\n";

            OpenFileId fid = kernel->machine->ReadRegister(6);
            //cout <<"fid = "<<fid<<"\n";

    int count;
    count = SysRead(buf, size, fid);

    kernel->machine->WriteRegister(2, (int) count);
            kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
            kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
            kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
            return;
            ASSERTNOTREACHED();
}
break;
```

Step 3. 修改/usrprog/ksyscall.h



```
int SysRead(char *buf, int size, OpenFileId fid)
{
    return kernel->fileSystem->ReadFile0(buf, size, fid);
}
```

Step 4. 修改/filesys/filesys.h

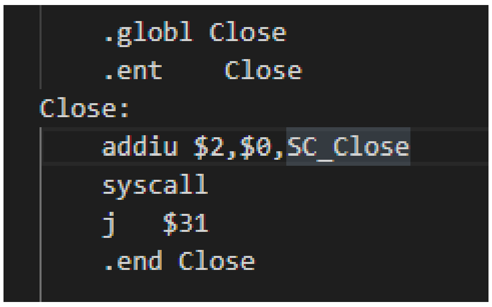


```
int ReadFile0(char *buf, int size, int fid){
Read(fid, buf, size);
return size;
}
```

Return size .

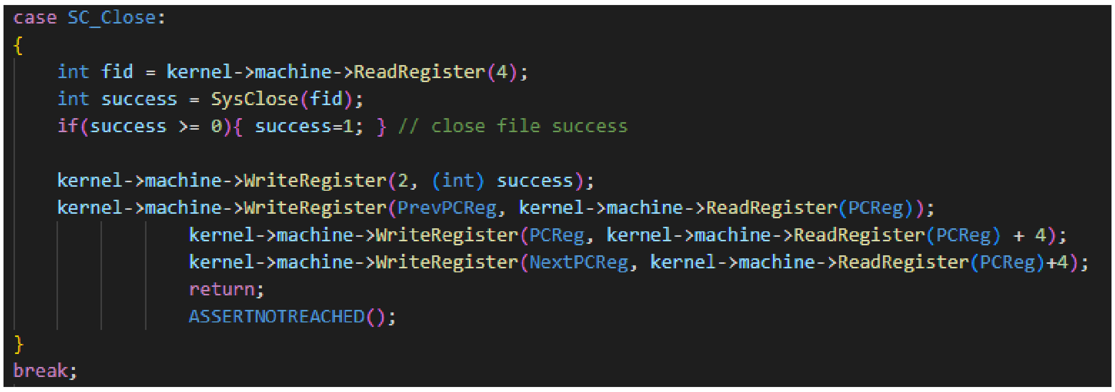
5. Close Systemcall Implement

Step 1. 修改/test/start.S



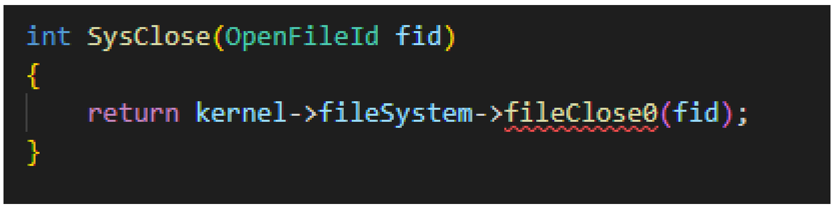
```
    .globl Close
    .ent    Close
Close:
    addiu $2,$0,SC_Close
    syscall
    j   $31
    .end Close
```

Step 2. 修改/usrprog/exception.cc



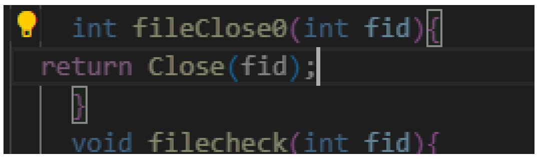
```
case SC_Close:
{
    int fid = kernel->machine->ReadRegister(4);
    int success = SysClose(fid);
    if(success >= 0){ success=1; } // close file success

    kernel->machine->WriteRegister(2, (int) success);
    kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
                kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
                kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
                return;
                ASSERTNOTREACHED();
}
break;
```

Step 3. 修改/usrprog/ksyscall.h



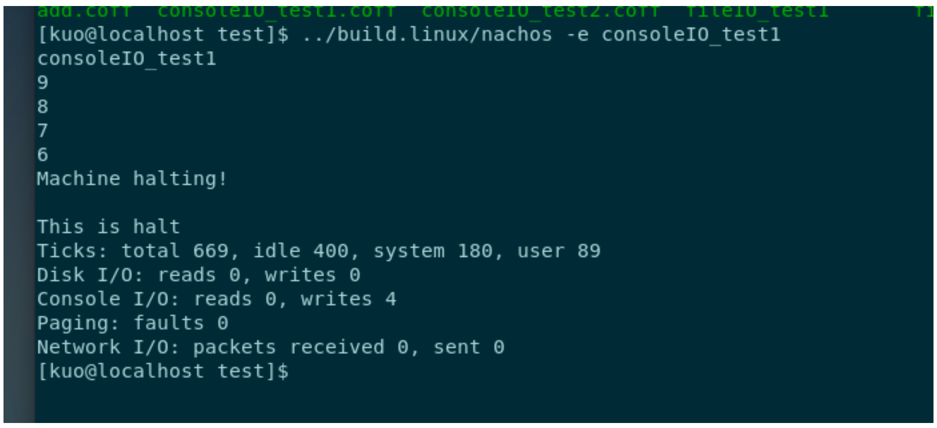
```
int SysClose(OpenFileId fid)
{
    return kernel->fileSystem->fileClose0(fid);
}
```

Step 4. 修改/filesys/filesys.h



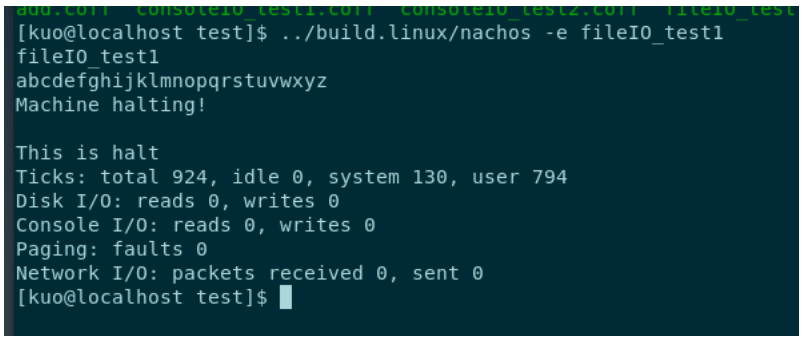
```
  int fileClose0(int fid){
return Close(fid);
  }
  void filecheck(int fid){
```

6. Requirement 1

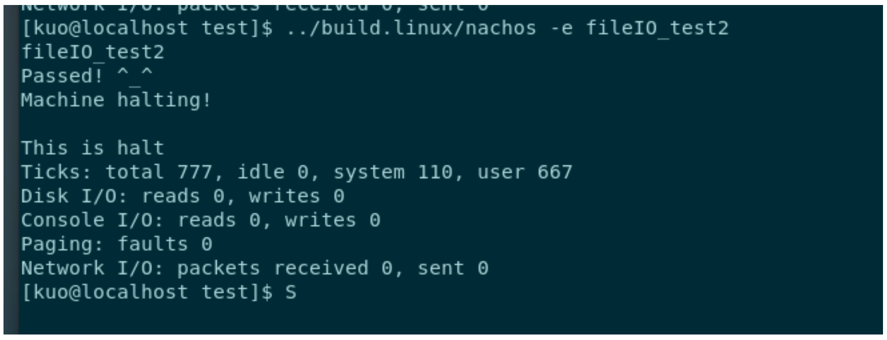


```
add.coff  consoleIO_test1.coff  consoleIO_test2.coff  fileIO_test1      fi
[kuo@localhost test]$ ../build.linux/nachos -e consoleIO_test1
consoleIO_test1
9
8
7
6
Machine halting!

This is halt
Ticks: total 669, idle 400, system 180, user 89
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 4
Paging: faults 0
Network I/O: packets received 0, sent 0
[kuo@localhost test]$
```

7. Requirement 2



```
add.coff  consoleIO_test1.coff  consoleIO_test2.coff  fileIO_test
[kuo@localhost test]$ ../build.linux/nachos -e fileIO_test1
fileIO_test1
abcdefghijklmnopqrstuvwxyz
Machine halting!

This is halt
Ticks: total 924, idle 0, system 130, user 794
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
[kuo@localhost test]$ 
```



```
Network I/O: packets received 0, sent 0
[kuo@localhost test]$ ../build.linux/nachos -e fileIO_test2
fileIO_test2
Passed! ^_^
Machine halting!

This is halt
Ticks: total 777, idle 0, system 110, user 667
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
[kuo@localhost test]$ S
```