FSS HW01 110598040

陳廷豪

Part 1 : Your implementation.

ConsoleIO_test1 and test2:

1. 打開userprog / syscall.h 去定義PrintInt()以及他function的宣告

syscall.h
~/Desktop/NachOS-4.0_MP1/code/userprog

```
#define SC_Create       4
#define SC_Remove       5
#define SC_Open         6
#define SC_Read         7
#define SC_Write        8
#define SC_Seek         9
#define SC_Close        10
#define SC_ThreadFork   11
#define SC_ThreadYield  12
#define SC_ExecV        13
#define SC_ThreadExit   14
#define SC_ThreadJoin   15
#define SC_PrintInt     16
#define SC_Add          42
#define SC_MSG          100
```

2. 到test / start.S新增MIPS code

start.S
~/Desktop/NachOS-4.0_MP1/code/test

```
ThreadJoin:
        addiu $2, $0, SC_ThreadJoin
        syscall
        j       $31
        .end ThreadJoin


/* dummy function to keep gcc happy */
        .globl  __main
        .ent    __main
__main:
        j       $31
        .end    __main


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

3. 到userprog / exception.cc的ExceptionHandler去新增SC_PrintInt這個case number

exception.cc
~/Desktop/NachOS-4.0_MP1/code/userprog

```
            kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
            }
            cout << "result is " << result << "\n";
            return;
            ASSERTNOTREACHED();
            break;
        case SC_PrintInt:
            val = kernel->machine->ReadRegister(4);

            char input[MAX];
            int len;
            len = new_value(val, input);
            kernel->synchConsoleOut->PutCharArray(input, len);

            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;
```

4. 由於預設的PutChar()都會把字串的'\n'算進來而導致write次數是原本預期的2倍，因此額外寫了一個PutCharArray()，讓我的write能show出正確的數字出來

5.先到userprog / synchconsole.cc與.h中去新增PutCharArray()的宣告與實作

synchconsole.h
~/Desktop/NachOS-4.0_MP1/code/userprog

```
    Semaphore *waitFor;         // wait for callBack

    void CallBack();            // called when a keystroke is available
};

class SynchConsoleOutput : public CallBackObj {
  public:
    SynchConsoleOutput(char *outputFile); // Initialize the console device
    ~SynchConsoleOutput();

    void PutChar(char ch);      // Write a character, waiting if necessary
    //add this new one, HW1
    void PutCharArray(char *ch, int len);
  private:
    ConsoleOutput *consoleOutput;// the hardware display
    Lock *lock;                 // only one writer at a time
    Semaphore *waitFor;         // wait for callBack

    void CallBack();            // called when more data can be written
};

#endif // SYNCHCONSOLE_H
```

Open synchconsole.cc ~/Desktop/NachOS-4.0_MP1/code/userprog

exception.cc syscall.h start.S synchconsole.h

```
//----------------------------------------------------------------------
// SynchConsoleOutput::CallBack
//      Interrupt handler called when it's safe to send the next
//      character can be sent to the display.
//----------------------------------------------------------------------

void
SynchConsoleOutput::CallBack()
{
    waitFor->V();
}


//add new one, HW1
void
SynchConsoleOutput::PutCharArray(char *ch, int len)
{
    lock->Acquire();
    consoleOutput->PutCharArray(ch, len);
    waitFor->P();
    lock->Release();
}
```

6.接著再到在machine中的console.cc與.h中去新增PutCharArray()

Open console.h ~/Desktop/NachOS-4.0_MP1/code/machine Save

exception.cc syscall.h start.S synchconsole.h synchconsole.cc

```
                                // initialize hardware console output
  ~ConsoleOutput();             // clean up console emulation

  void PutChar(char ch);        // Write "ch" to the console display,
                                // and return immediately.  "callWhenDone"
                                // will called when the I/O completes.
  void CallBack();              // Invoked when next character can be put
                                // out to the display.

  //add this new one, HW1
  void PutCharArray(char *ch, int len);


private:
  int writeFileNo;                      // UNIX file emulating the display
  CallBackObj *callWhenDone;            // Interrupt handler to call when
                                        // the next char can be put
  bool putBusy;                         // Is a PutChar operation in progress?
                                        // If so, you can't do another one!
};
```

console.cc
~/Desktop/NachOS-4.0_MP1/code/machine

exception.cc | syscall.h | start.S | synchconsole.h | synchconsole.cc | console.h

```
//      to occur in the future, and return.
//----------------------------------------------------------------------

void
ConsoleOutput::PutChar(char ch)
{
    ASSERT(putBusy == FALSE);
    WriteFile(writeFileNo, &ch, sizeof(char));
    putBusy = TRUE;
    kernel->interrupt->Schedule(this, ConsoleTime, ConsoleWriteInt);
}


//Add this new one, HW1
void
ConsoleOutput::PutCharArray(char *ch, int len)
{
    ASSERT(putBusy == FALSE);
    WriteFile(writeFileNo, ch, sizeof(len));
    putBusy = TRUE;
    kernel->interrupt->Schedule(this, ConsoleTime, ConsoleWriteInt);
}
```

7. 輸出結果

```
[teigou@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

[teigou@localhost test]$ ../build.linux/nachos -e consoleIO_test2
consoleIO_test2
15
16
17
18
19
Machine halting!

This is halt
Ticks: total 826, idle 500, system 220, user 106
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 5
Paging: faults 0
Network I/O: packets received 0, sent 0
```

FileIO_test1 and test2:

1.因為test / start.S還有userprog / syscall.h中早已經有定義好這四個的MIPS code和function宣告了所以跳過

2.直接到exception.cc的ExceptionHandler去自己新增SC_Open, SC_Close, SC_Read, SC_Write的實作內容, 這樣就完成了

```
switch (which) {
case SyscallException:
            switch(type)
            {
            case SC_Open:
                    val = kernel->machine->ReadRegister(4);
                    {
                            char *filename = &(kernel->machine->mainMemory[val]);
                            status = OpenForReadWrite(filename, false);
                            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;

 case SC_Read:
         {
         val = kernel->machine->ReadRegister(4);
         char *buffer = &(kernel->machine->mainMemory[val]);
         int size = (int)kernel->machine->ReadRegister(5);
         file = new OpenFile((int)kernel->machine->ReadRegister(6));
         int count = file->Read(buffer, size);
         status = (count == size) ? size : -1;
         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;

case SC_Write:
        {
        val = kernel->machine->ReadRegister(4);
        char *buffer = &(kernel->machine->mainMemory[val]);
        int size = (int)kernel->machine->ReadRegister(5);
        file = new OpenFile((int)kernel->machine->ReadRegister(6));
        int count = file->WriteAt(buffer, size, file->Length());
        status = (count == size) ? size : -1;
        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;
```

```
case SC_Close:
        val = kernel->machine->ReadRegister(4);
        {
        status = (Close(val) == 0) ? 1 : 0;
        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;
```

3.不過在測試時好像有小問題，就是有時候執行FileIO_test會秀出failed open的error msg，這部分要一直不斷地重新在build.linux裡面make clean->make，再到test裡面make clean->make然後重新執行幾遍同樣的步驟才有機會解決，時好時壞。

4.輸出結果

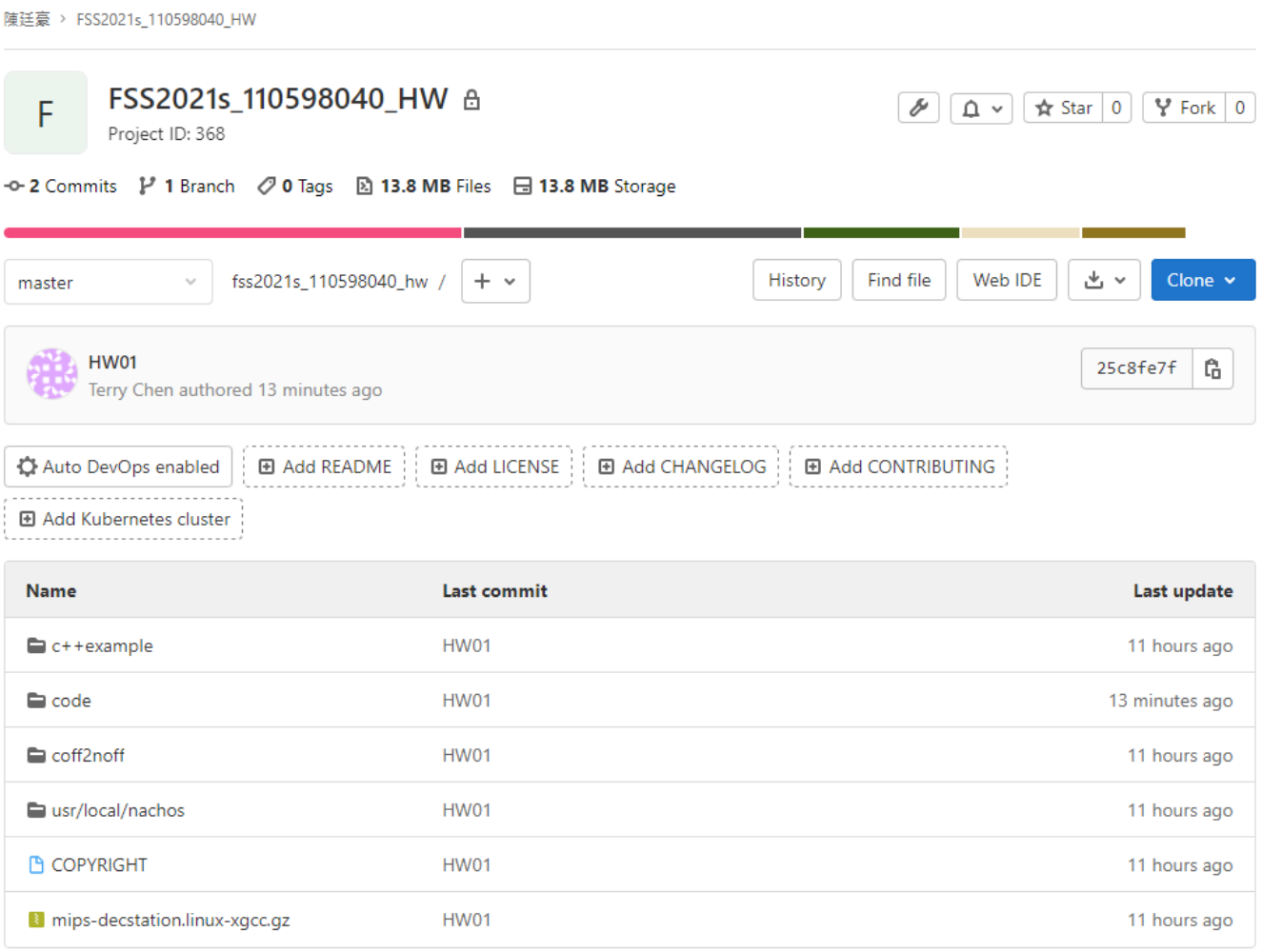
```
[teigou@localhost test]$ ../build.linux/nachos -e fileIO_test1
fileIO_test1
Machine halting!

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
```

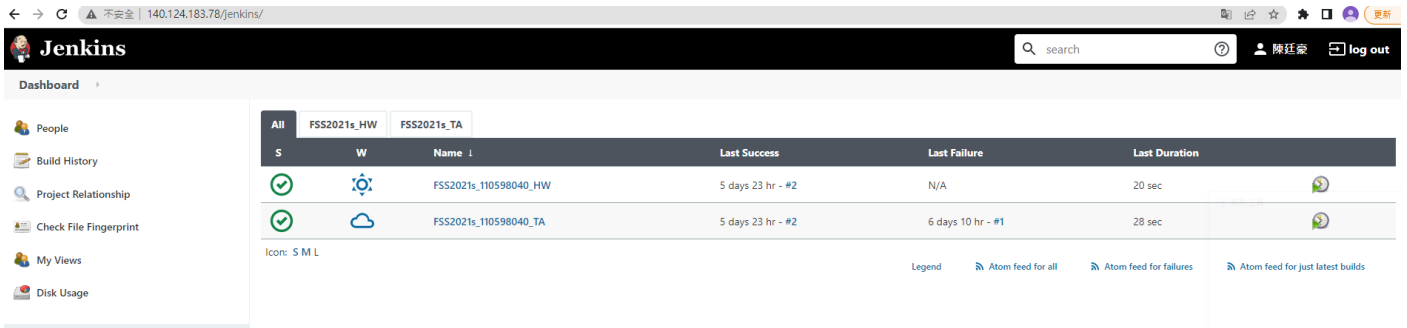
```
[teigou@localhost test]$ ../build.linux/nachos -e fileIO_test2
fileIO_test2
Passed! ^_^
Machine halting!

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
```

Gitlab:



Jenkins:



Console Output(TA):

```
[32m[ RUN      ][0m HW00_halt
[32m[       OK ] [0mHW00_halt
[32m[ RUN      ][0m HW01_Console_test1
[32m[       OK ] [0mHW01_Console_test1
[32m[ RUN      ][0m HW01_Console_test2
[32m[       OK ] [0mHW01_Console_test2
[32m[ RUN      ][0m HW01_File_test1
[32m[       OK ] [0mHW01_File_test1
[32m[ RUN      ][0m HW01_File_test2
[32m[       OK ] [0mHW01_File_test2
[32m[ RUN      ][0m HW01_File_test3
[32m[       OK ] [0mHW01_File_test3
[32m[ RUN      ][0m HW01_File_test4
[32m[       OK ] [0mHW01_File_test4
[32m[  PASSED  ][0m 7 tests.
Points:  100 / 100
Started calculate disk usage of build
Finished Calculation of disk usage of build in 0 seconds
Started calculate disk usage of workspace
Finished Calculation of disk usage of workspace in 0 seconds
Finished: SUCCESS
```