**CS342301: Operating System**

**MP1: System Call**

**Team31**

**111060013 劉祐廷 111060019 黃子恩**

|  |  |  |
| --- | --- | --- |
|  | **劉祐廷** | **黃子恩** |
| **Contribution** |  |  |

1. **Trace Code**
   1. **SC\_Halt**

To run a user-level program, turn into usermode. Execute instructions one by one calling Machine::OneInstrution().

* + 1. **Machine::Run()**

This function is responsible for simulating the process of the CPU executing a program in NachOS. It first sets the mode to user mode by calling kernel->interrupt->setStatus(UserMode);. Then, it enters an infinite loop, for(;;){}, which simulates the CPU's continuous fetching of instructions during execution. Within the loop, the OneInstruction() function is called to execute an instruction. After each execution of OneInstruction(), the OneTick() function is invoked to check for and handle any interrupts that may have occurred.

* + 1. **Machine::OneInstruction()**

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

First, it fetches the instruction from registers[PCReg]. If it cannot fetch an instruction, it returns; if an instruction is fetched, it proceeds to decode it.一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Next, it uses the opCode to determine the action represented by the instruction.一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Using the action OP\_ADD as an example, if an exception occurs, the error code is passed as a parameter to RaiseException() for handling. If no exception occurs, the respective operations are performed as usual.

* + 1. **Machine::RaiseException()**



The problematic virtual address is stored in the register designated for holding the bad virtual address.

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

To handle the exception, turn into kernel mode. Call the ExceptionHandeler(), then turn into usermode again.

* + 1. **ExceptionHandler()**

一張含有 文字, 字型, 螢幕擷取畫面, 數字 的圖片

自動產生的描述

From start.S, it can be observed that the codes for these exceptions are all stored in the r2 register.



Therefore, the system call code can be obtained by reading the value of r2, as the system call code is stored in r2.

一張含有 字型, 文字, 圖形, 螢幕擷取畫面 的圖片

自動產生的描述

First, check whether the system call code exists.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Next, perform different actions based on the system call code. Since the system call code passed in is SC\_Halt, the program will execute this section of code to call SysHalt().

* + 1. **SysHalt()**



Call the Halt() function defined in the kernel's interrupt operations.

* + 1. **Interrupt::Halt()**

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Deconstruct the entire kernel, releasing resources and stopping the entire system.

* 1. **SC\_Create**

1. **ExceptionHandler()**

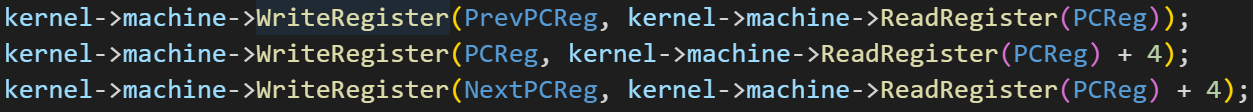


Read the address of the file name string to be added from r4 (which is usually arg1).

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Extract the filename and pass it to SysCreate() to create the file, then write the return value to r2.



Update the PC to prepare for executing the next instruction.

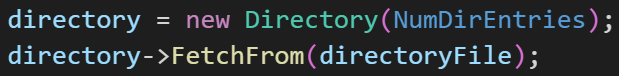
1. **SysCreate()**

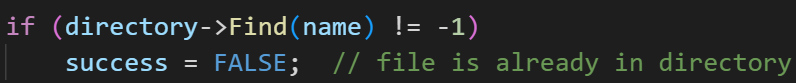
一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Call the Create() function in the fileSystem and pass the filename to it.

1. **FileSystem::Create()**



Obtain the corresponding directory.

Check if the file exists. If it does, then return false.一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Check if there is enough space to create the file. If not, then return false.一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Try to allocate space for the file. If there is not enough space, then return false.一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Successfully create the file and write the modified data back to the disk.一張含有 文字, 字型, 螢幕擷取畫面, 圖形 的圖片

自動產生的描述

Deconstruct these pointers to release memory resources.

* 1. **SC\_PrintInt**
     1. **ExceptionHandler()**

一張含有 文字, 字型, 螢幕擷取畫面, 數字 的圖片

自動產生的描述

From start.S, it can be observed that the codes for these exceptions are all stored in the memory location referenced by r2.



Therefore, the system call code can be obtained by reading the value of r2, as the system call code is stored in r2.

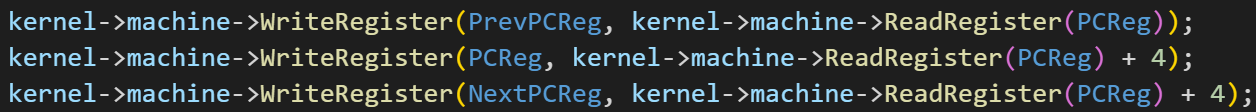
一張含有 字型, 文字, 圖形, 螢幕擷取畫面 的圖片

自動產生的描述

First, check whether this system call code exists.一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Next, perform different actions based on the system call code. Since the passed system call code is SC\_PrintInt, the program will execute this section of code. First, it will read the value to be output from r4 (which is arg1) and store it in val. Then, it will pass val as a parameter to SysPrintInt().



Update the PC to prepare for executing the next instruction.

* + 1. **SysPrintInt()**

一張含有 文字, 字型, 螢幕擷取畫面, 印刷術 的圖片

自動產生的描述

Pass the value to be output (val) to the PutInt() function, which is already defined in the kernel's synchConsoleOut.

* + 1. **SynchConsoleOutput::PutInt()**

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

Convert the number to a string and use a loop to pass each character one by one to PutChar().

* lock->Acquire();

This line of code acquires a lock before performing operations on shared resources, ensuring that the current process has exclusive access to the resource before entering the critical section. This prevents other processes from simultaneously accessing consoleOutput, which could lead to mixed output data. When the lock is held by one process, any other processes attempting to acquire the lock will be blocked until the lock is released.

* lock->Release();

Once this operation is complete, the lock is released, allowing other processes to enter the critical section. This ensures that after a complete output operation has finished, other processes can safely execute their own output operations.

* + 1. **SynchConsoleOutput::PutChar()**

Compared to SynchConsoleOutput::PutInt(), it simply omits the process of converting the number to a string; the remaining parts are largely the same.

* + 1. **ConsoleOutput::PutChar()**



putBusy indicates whether the console output is busy. When it is FALSE, it means the console is idle, and a new output can begin. If it is TRUE, it indicates that an output operation is in progress, and this call should be blocked to prevent multiple characters from being output simultaneously.

Simulate outputting ch to the console output.



Set putBusy to TRUE to indicate that the console output is busy, preventing multiple characters from being output simultaneously.

Call Schedule() to raise an interrupt after the output is completed, indicating that the output operation is finished.

* + 1. **Interrupt::Schedule()**



Calculate the time at which this interrupt should be triggered.一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

Construct a PendingInterrupt and check if fromNow is greater than 0 (indicating an event that will occur in the future). If it is, add this PendingInterrupt to the sorted list, waiting for it to be triggered.

* + 1. **Machine::Run()**

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Run() will call OneTick() after each invocation of OneInstruction(). The purpose is to check for any interrupts after executing each instruction and to update the time ticks.

* + 1. **Machine::OneTick()**



Save the current state for data recovery purposes.

Obtain the statistics metrics to update the ticks.一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

Update the ticks according to the current mode.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

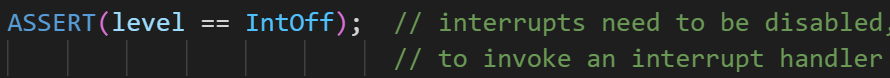
自動產生的描述

Call ChangeLevel() to disable interrupts, allowing the interrupt handler to execute. Then, call CheckIfDue() to check if any PendingInterrupts are due and need to be raised. Finally, call ChangeLevel() again to re-enable interrupts.一張含有 文字, 字型, 螢幕擷取畫面, 筆跡 的圖片

自動產生的描述

Check if the timer has requested a context switch. If so, switch directly to the new context. Since yield is managed by the kernel, it is necessary to first switch to system mode, allowing the CPU to change tasks. After the switch is complete, restore the status to continue execution.

* + 1. **Interrupt::CheckIfDue()**



Ensure that interrupts are disabled.一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

If there are no interrupts pending, simply return FALSE.



Read the next interrupt that is scheduled to be executed.

If it is not yet time for the interrupt to occur.一張含有 字型, 文字, 螢幕擷取畫面, 圖形 的圖片

自動產生的描述

If it is not yet time to trigger the interrupt, return FALSE.

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

When advanceClock is true, it indicates that there are no other processes ready at the moment, allowing the system to advance the time to this interrupt. This will also increase the idle ticks, so idleTicks should be updated as well.

一張含有 文字, 字型, 螢幕擷取畫面, 筆跡 的圖片

自動產生的描述

If a machine simulator exists, a delay load must be executed, which is a step between the kernel and the hardware.

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Set inHandler to TRUE, indicating that an interrupt is being handled. Use a while loop to check if any interrupts in the pending list have expired; if they have, remove them and call the appropriate CallBack() from callOnInterrupt based on the next interrupt type to handle the interrupt. Finally, after the interrupt has been processed, set inHandler back to FALSE, indicating that the interrupt handling is complete.

* + 1. **ConsoleOutput::CallBack()**

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

When a character output is completed, putBusy will be set to FALSE, indicating that the next character can now be output. Update the count of successfully output characters, then call the CallBack() from callWhenDone to notify the system that the next character can be output.

* + 1. **SynchConsoleOutput::CallBack()**

一張含有 文字, 字型, 螢幕擷取畫面, 筆跡 的圖片

自動產生的描述

After the output is complete, use waitfor's V() to notify the system to release the lock, allowing the next process to output. Compared to ConsoleOutput::CallBack(), this includes an additional locking mechanism to ensure the correctness of output in a multiprocess environment.

* 1. **Makefiles**

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

Using halt as an example, when you type make halt in the terminal, it will jump to the halt label and check the files following the colon. It will then recursively go through each file, ensuring that they are all properly processed.…

CFLAGS is a variable that stores a set of directives to inform the compiler which options to use during the compilation process.

* -g：Let the compiler to generate debugging information.
* -G 0：-G is typically used in MIPS architecture compilers, where 0 indicates the size of variables placed in the global pointer area (GPA), and 0 signifies that no optimization is performed.
* -c：Tell the compiler to compile only the source files without performing the linking process, which will produce .o files.
* $(INCDIR)：This is a variable that stores the path to the header files. When the compiler encounters #include, it will search for the header files in this path.
* -B/usr/bin/…：-B makes the following paths binary, allowing the compiler to search for tools like the linker in those paths.

The CC variable represents the path to the gcc program (compiler).The LD variable represents the path to the ld program (linker).This line is responsible for creating halt.o from halt.c.一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

This segment of code is responsible for linking halt.o and start.o together to create halt.coff. The .coff file is executable on MIPS architecture, but since the Linux architecture is x86, it needs to be converted to .noff to become a file that can be executed on the MIPS simulator.

1. **Implementation of I/O System Calls in NachOS**
2. **Difficulties**
3. **Feedback**