# CMPS 455

October 13, 2021

**Project 3 - System Calls and Exceptions Notes**

From Jason Woodworth's *System Calls and Exceptions in NachOS*

## ../code/test

- system calls (exposes kernel functionality)
- Halt.c, etc.
-Exec() executes user program, returns threadID it runs
-Join() takes an Exec() arg -→ Join(Exec("path/to/program"))
  - puts current thread to sleep
  - waits til new thread that it was passed finishes, then wakes up old thread

## ../code/machine/mipssim.cc

-ReadMem()
  -read instruction
  -might have exception
    -RaiseException()
      -OverFlowException (adding numbers etc)
      -AddressErrorException (invalid address)
      -SysCallException
-machine.h
  -ExceptionType( NoException,
    SysCallException,
    PageFaultException, (need to load new page)
    ReadOnlyException,
    BusErrorException,
    AddressErrorException, (reading wrong things into mainMemory,
      when doing read add into mainMemory)
    OveflowException,
    IllegalInstrException,
    NumExceptionTypes )
  -deletes currentThreads→space
  -currentThread→Finish()
  -need to add another case for PageFaultException

- ../userprog/exception.cc
  - ExceptionHandler(ExceptionType which)
    - arg_n = machine→ReadRegister(int);
    - syscall, switch on type, stored in register 2
    - SC_Halt
    - SC_Read
    - SC_Write, debug your own userprog
    - SC_Exec
      - returns address of filename (arg1)
      - open file as executable
      - set AddrSpace(executable)
      - check if there is enough space
        - if !currentThread→killNewChild (not enough memory)
        - else, new Thread
          - set space, id, add to list of active threads
          - write register2 of its threadID (return value of syscall)
          - threadID++ (global)
          - execThread→Fork(processCreator, 0) //it gets forked
          - This is all very similar to StartProcess (in progtest.cc)
            - processCreator(int arg)
              - inits register, restores state, checks if threadToBeDestroyed (calling thread)
              - machine→Run() (sets off inf loop for program)
    - SC_Join
      - join one process to another
      - arg is threadID, from Exec() that we want to Join() as a child
      - Join()→Exec()→ThreadID==1 (because main was 0)→0 is now parent of 1→0 goes to sleep while 1 runs→once 1 stops 0 wakes back up
      - Exec() will return negative number if thread was not properly created (throws error)
      - if thread does exist (by calling getID) and not joined by another process (should be asleep, cant join while asleep)
        - Join() process with new process
        - setParent(currentThread)
        - currentThread→isJoined = true
        - currentThread→Sleep(); break;
        - in threads destructor, it will tell parents to wake up
    - SC_Exit
      - exit process, checks if normal exit, delete currentThread space, Finish()
    - SC_Yield
      - save state of process
      - restore state of process