Dylan McQuaid

421 Project 2 Initial Design Document

System Calls to Implement:

1. long sbx421_block(pid_t proc, unsigned long nr)
    a. SYSCALL_DEFINE2(sbx421_block, pid_t, proc, unsigned long, nr)
    b. proc is process id to be blocked
    c. nr is identifying number of system call process is to be blocked from
2. long sbx421_unblock(pid_t proc, unsigned long nr)
    a. SYSCALL_DEFINE2(sbx421_unblock, pid_t, proc, unsigned long, nr)
    b. proc is process id to be unblocked
    c. nr is identifying number of system call process is to be unblocked from
3. long sbx421_count(pid_t proc, unsigned long nr)
    a. SYSCALL_DEFINE2(sbx421_count, pid_t, proc, unsigned long, nr)
    b. proc is process id to be unlocked
    c. nr is identifying number of system call to get count of how many times the process identified by proc tried to call the system call when blocked by it

436 system calls after adding block, unblock and count at 434, 435, and 436 respectively

Implementation:

- Project will be implemented using skiplist code from project 1
- Being that the number of system calls is fixed at 436 (could increase if more added), will create an array of the system calls, where each index 0-436 represents the number of the system call, and each system call has its own skiplist
    - static struct SkipList *syscalls[436 + 1]
    - 436 will be defined as a global NUM_OF_SYSCALLS
    - MAX_LEVEL for the skiplist is obtained by the max number of PIDS possible.
    - Rough implementation of block code in user space prototype is included below. Still needs error checking and more, but is functional

```c
struct SkipNode{
  unsigned int process_id;
  unsigned int count;
  struct SkipNode **forward;
};

struct SkipList{
  unsigned int CurrentLevel;
  unsigned int MaxLevel;
  unsigned int LevelProb;
  struct SkipNode *header;
};

#define NUM_OF_SYSCALLS 436
#define PROB 2
#define MAX_LEVEL 32768
#define MAX_ID UINT_MAX;

static struct SkipList *syscalls[NUM_OF_SYSCALLS + 1];
```

```c
long sbx421_block(pid_t proc, unsigned long nr){
  if(syscalls[nr] == NULL){
    printf("IN HERE\n");
    init_syscall_list(nr);
  }

  struct SkipNode *update[syscalls[nr]->MaxLevel + 1];
  struct SkipNode *x = syscalls[nr]->header;
  int level, i;

  for(i = syscalls[nr]->CurrentLevel; i >= 1; i--){
    while(x->forward[i]->process_id < proc){
      x = x->forward[i];
    }
    update[i] = x;
  }

  x = x->forward[1];

  if(x->process_id == proc){
    printf("Process is already blocked by this system call\n");
    return -1;
  }
```

```c
  else{
    level = get_random_level();
    if(level > syscalls[nr]->CurrentLevel){
      for(i = syscalls[nr]->CurrentLevel + 1; i <= level; i++){
        update[i] = syscalls[nr]->header;
      }
      syscalls[nr]->CurrentLevel = level;
    }

    x = malloc(sizeof(struct SkipNode));
    x->process_id = proc;
    x->forward = malloc(sizeof(struct SkipNode*) * (level + 1));
    for(i = 1; i <= level; i++){
      x->forward[i] = update[i]->forward[i];
      update[i]->forward[i] = x;
    }
  }

  return 0;
}
```

Changes to be Made:

- Inside /usr/src/project2/Makefile
    - On line 996 add after “block/” “proj2/kernel/”
- Inside /usr/src/project2/include/linux/syscalls.h
    - At the bottom of the file (right before the endif) add a line for each system call
    - asmlinkage long sys_sbx421_block(pid_t proc, unsigned long nr);
    - asmlinkage long sys_sbx421_unblock(pid_t proc, unsigned long nr);
    - asmlinkage long sys_sbx421_count(pid_t proc, unsigned long nr);
- Inside /usr/src/project2/arch/x86/entry/syscalls/syscall_64.tbl
    - Add the lines:
    - 434 common sbx421_block …
    - 435 common sbx421_unblock …
    - 436 common sbx421_count …
- Inside /usr/src/project2/
    - Add README describing the system calls and providing any references used
- Inside /usr/src/project/proj2/design
    - Add .pdf files of initial design and final design
- Inside /usr/src/project2/proj2/kernel
    - Add proj2code.c file where the 3 system calls will be implemented
    - Add Makefile to compile proj2code.c
- Inside /usr/src/project2/proj2/misc
    - Add prototype proj2proto.c to test implementation of system calls in user space
    - Add README describing prototype code
- Inside /usr/src/project2/proj2/testing
    - Add test files using the system calls
- Inside /usr/src/project2/
- Edit entry, so that when a system call is made the syscalls array above is accessed using that system calls number and the skiplist is checked to see if that process is on that

system calls blocked list, if it is update the count variable for that SkipNode that represents that process_id

- This change to be made inside /usr/src/project2/arch/x86/entry/common.c
- The code will be added inside do_audit_syscall_entry() or syscall_trace_enter()
- regs->orig_ax represents the system call number so would use that to access the syscalls array. Like follows syscalls[regs->orig_ax]
- (not sure exact lines yet, but I believe I am on the right track with one of these two functions