CSL373/CSL633 Minor 1 Exam
Operating Systems
Sem II, 2014-15

Answer all 6 questions Max. Marks: 32

Unix System Calls

1. Write the pseudo-code for a program 'cp' that takes two command-line arguments, say input-file and output-file, and copies the input file to the output file. [3]

Syntax:
$ cp ifile ofile

Code for cp:

```
void main(int argc, char **argv)
{
   // your solution goes here. use UNIX system calls to implement the functionality.

  char buf[SIZE];
  assert(argc>2);
  ifd = open(argv[1], O_READONLY); // open inputfile in readonly mode
  ofd = open(argv[2], O_WRITEONLY  // open outputfile in writeonly mode
                | O_CREAT      // O_CREAT: if outputfile doesn't exit.
                | O_TRUNC);    // O_TRUNC: If outputfile size >
inputfilesize

  for(;;){
    ibytes = read(ifd, buf, sizeof(buf));
    if(ibytes <= 0) {
      break;  // no need to handle EAGAIN case.
    }

    obytes = write(ofd, buf, ibytes);
    assert(obytes == ibytes); // no need to handle partial writes
  }

  close (ifd);
  close (ofd);
}
```

2. Consider the following function:

```
int32_t global;

int32_t foo(int32_t a, int32_t *b) {
  int32_t c;

  c = global + a;

  return *(b + c);
}
```

Assume that the variable global is allocated at a global address 0x12345. Write the assembly code for this function, with proper comments on which assembly code lines are implementing which C statement. Assume GCC calling conventions. You will need to be careful about properly naming all variables and arguments (e.g., using global addresses, stack offsets or frame pointer offsets), use proper opcodes and addressing modes, obey caller and callee-save conventions, etc. It is okay to not be exactly correct in the use of x86 opcodes, but the general layout of the code and its logic should be correct. [6]

```
> int32_t global;
> int32_t foo(int32_t a, int32_t* b){

.text
foo:
    pushl %ebp
    movl  %esp, %ebp

> int32_t c;

    subl $4, %esp
    # Alt: pushl $0
    # Alt: register allocate c, say %ecx

> c=global+a;

    movl global, %ecx
    addl 8(%ebp),%ecx

    #Alt:
    #  movl 8(%ebp),%ecx
    #  addl global, %ecx
```

```
> return *(b+c);

    # In C, *(b+c) means: *(int32_t*)(uintptr_t(b)+ c*sizeof(int32_t))
    # movl 12(%ebp), %ebx
    # movl (%ebx,%ecx,4), %eax # eax = *(ebx+ecx*4)

    #Alt:
    # movl  12(%ebp), %ebx
    # shll  $2, %ecx    # imull $4, %ecx
    # addl  %ecx, %ebx
    # movl  (%ebx), %eax

   # return value on %eax

> }
```

Notes for TAs to ease evaluation:

Syntax for eax=base+index*scale+disp:
AT&T : movl disp(base,index,scale), %eax
Intel: mov %eax,[base+index*scale+disp]

With prelude:
EBP+12 : &b
EBP+8 : &a
EBP+4 : ret
EBP+0 : old_ebp
EBP-4 : &c = ESP+0

Without prelude:
ESP+12 : &b
ESP+8 : &a
ESP+4 : ret
ESP+0 : &c

Without prelude and register allocated c:
ESP+8 : &b
ESP+4 : &a
ESP+0 : ret

3. How does the OS ensure through segmentation that one application cannot access another application's address space in the following situations: [10]

A. The application tries to write to the physical address of the other application.
B. The application tries to modify the segment register
C. The application tries to overwrite GDT entries
D. The application tries to lower its privilege-level (i.e., tries to gain supervisor privileges).

A. The Application tries to write to the physical address of the other application:

- Base+Limit of each entry in GDT table.
  Range of all code/data entries in GDT with DPL=3 is subset of user's address space.
- Application's CPL=3.
- Every app's address space is mutually exclusive.

B. The application tries to modify the segment register

- Base+Limit in GDT's descriptor register when executing lgdt
  So application cannot load random descriptor.
- H/w: Type checking: If application tries to load Trap gate etc in GDT entires as CS or DS etc: will trigger fault
- Application's CPL is 3.
- Application can modify it's segment register to any desriptor with DPL>=CPL, ie. DPL=3
  Range of all code/data entries in GDT with DPL=3 is subset of user's address space.

C. The application tries to overwrite GDT entries

- Intersect(GDT,application's address space) = non empty

D. The application tries to lower its privilege level - (ring0)

Application's CPL=3.
Application can change it's PL by gates.

Call gates and Trap gates:
call gates and trap handlers' entry points are defined by OS.
Even though privilege level is reduced while executing trap handlers,
OS will ensure that it'll be restored back when the trap handler returns.

4. The instruction to load the Interrupt Descriptor Table Register (IDTR) is “lidt” and is a privileged instruction, i.e., it can only be executed in privileged mode. Assume that it was possible to execute this instruction in user mode by an untrusted user process. Show an attack using this additional (hypothetical) capability, whereby:

A. A user process can crash the machine.
B. A user process can read the memory contents of another process.

You should show the steps that the user process should follow to launch this attack in as much detail as possible. [8]

A. User process can crash the machine:

Criteria: Control shouldn't reach OS.

Solution 1:
1. new local structure of IDT
   - all elements filled with valid user's function.
2. Take control: infinite loop

Solution 2:
1. new local structure of IDT
   - all elements filled with valid user's function which will do infinite loop
2. Take control: Generate trap

Solution 3:
1. new local structure of IDT
   - all elements filled with zero
2. Take control: Generate trap to get triple fault

Note: If user didn't set valid flags, say DPL=3, or Present: 0. implies triple fault. So is a correct solution

B. User process can read the memory contents of another process:

0. Save current idt pointer.
1. new local structure of IDT
   - all elements filled with valid user's function
   - Valid Flags: DPL field should be 3. Present: 1
2. Set stack pointer to a valid memory, not to cause stack overflow case
3. Trigger traps: by generating traps like division by zero, executing int $0 etc.

Now user's function starts executing in ring 0.

In user's function:
- cli / prevent nested by using a variable (optional)
- Make a valid entry in our page table and make it point to target app's physical address.
  - Note: Using our existing Page table entry, we can't access target app's address space
  - User need to take care of following cases:
    - Limits in Segment registers in GDT.
    - If invalid entry is adding or a new page table, we need to reset segment registers as well
- copy the data to our address space
- Revert the page table modification
- iret

From the app: restore saved idt

Assumption: LIDT Takes linear address as it's input: So mechanism for computing linear address exists

5. Assume a memory access latency of 100ns, and a 2-level page table hierarchy on 32-bit x86. What should be the TLB hit rate to ensure that the average memory access latency is 102ns. Assume there are no instruction/data caches in the hardware. [4]

Assumption: TLB contains VA->PA only.

If intermediate mapping not considered:

Hit = 100
Miss = 300
Weighted avg of 100 and 300= 102

Solving eq:

(1-mm)*100 + mm*300 = 102
100 + mm*(300-100) = 102
mm*200 = 2
mm = 0.01 : 1%

or hit rate = 1-mm = 99%

If intermediate mapping is considered:

Hit = 100
Intermediate = 200
Miss = 300
Weighted avg of 100, 200 and 300= 102

Solving eq:

(1-mi-mm)*100 + mi*200 + mm*300 = 102
100 + (200-100)*mi + mm (300 -100) =102
100*mi + 200*mm = 2
mi + 2*mm = 0.02

If mi = 0. then mm = 0.01 => 99%, 0%, 1%
If mi = 0.01, then mm = 0.005 => 98.5%, 1%, 0.5%

6. Currently, the physical OS lectures repeat much material that is already present in the video lectures (let's call the current format, format A). We are contemplating a new format (say format B), where we assume that you have already listened to the corresponding video lecture before coming to the physical lecture, and spend the physical lecture time in quickly reviewing the material, question-answer session, exercise-solving, and most importantly, discussing advanced concepts around that material. (We believe that many students already listen to video lectures in advance). This way, we think you can learn much more from this course. Note that the "advanced concepts" will not be a part of exam syllabus.

Would you prefer format A or format B? Why? Please be honest in your answer.

For example, answers like "I am neutral" or "I don't care because I will not attend physical lectures in either case", are perfectly valid honest answers and will fetch full marks for this question. An answer like "I would love to learn more advanced concepts, and so I like format B" is also a valid answer.

We will plan the future lectures of this course based on this survey. [1]