Operating Systems and C : Exam

Willard Rafnsson & Niclas Hedam, IT University of Copenhagen, Fall 2023

This is the hand-out for the exam for Operating Systems and C, Fall 2023.
Your course grade will be based on your answers to the questions below.

clab (10 points)

(a)

Suppose, in an otherwise working implementation of **clab**, `queue_new` were defined as follows.

C

```c
/* @return The new queue, or NULL if memory allocation failed */
queue_t *queue_new(void) {
    return malloc(sizeof(queue_t));
}
```

This compiles, and satisfies the requirement stated in the comment.
However, this `queue_new` is incorrect. Explain why.
Hint: What did we neglect to do? How is that a problem? How do we fix that?
Hint: running `qtest` on `trace.cmd` consisting of

```
new
ih hello_world
reverse
rh world
```

produces the following output.

SHELL

```
$ ./qtest -v 2 -f trace.cmd
cmd> new
cmd> ih hello_world
cmd> reverse
[1]    6015 segmentation fault (core dumped)  ./qtest -v 2 -f trace.cmd
```

(b)

Describe your implementation of `queue_free`.
Hint: explain each step, and why it is needed.

asmlab (10 points)

(a)

Describe your solution to Part I (binary bomb) Phase 3.
Hint: what input defuses this phase? how did you come by this input: where is it stored? why does it have this form? what do the compare-instructions compare? explain your process (objdump, gdb, etc.)

(b)

Describe your attack in Part II (buffer overflow) Phase 4.
Hint: what is your payload? what is its overall structure? what do you wish to accomplish? how does your payload accomplish this?

prflab (10 points)

(a)

Describe your implementation of `rotate_t`.

Hint: show your implementation, say what each part does, then explain how your implementation came to be: which program optimizations yielded speedups, which did not, and why? how did you split the whole task into subproblems? what do the `pthreads` function calls do?

(b)

Would vectorization benefit `rotate_t` ? How, or why not?

Would multithreading benefit `blend_v` ? How, or why not?

Hint: What is it about the problem that makes this (not) beneficial? For a "why not", give a solid argument. For a "how", demonstrate the benefit.

syslab (10 points)

(a)

Describe your implementation of multithreading.

Hint: how do you ensure that the proxy is always ready to receive requests? which part of the request-processing do you hand off to a thread?

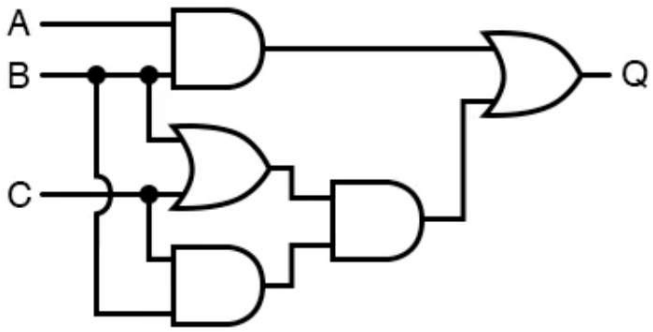
(b)

Describe your implementation of caching.

Hint: what data structure do you use to represent the cache? how do you prevent race conditions?

logic (6 points)

Here is a logic diagram.



(a)

Which of the following Boolean expression are equivalent with Q (the output of the circuit)?

- A&B | B&C & B&C
- A&B&C
- A&B | B&C&(B|C)
- B&(A|C)

Pick all that apply.

(b)

Which of the following are entries in the truth table for this circuit?

- A=1, B=0, C=1, Q=0
- A=1, B=1, C=0, Q=1
- A=0, B=1, C=1, Q=1
- A=1, B=1, C=0, Q=0

Pick all that apply.

assembly (9 points)

Here is a short program written in Y86-64.

```x86asm
  irmovq $D, %rdi      # instruction nr. 0
  irmovq $S, %rsi      # instruction nr. 1
  irmovq two, %rax     # instruction nr. 2
  mrmovq 0(%rax), %rax # instruction nr. 3
main:
  addq %rdi, %rax      # instruction nr. 4
  addq %rax, %rax      # instruction nr. 5
  subq %rdi, %rsi      # instruction nr. 6
  je done              # instruction nr. 7
  jmp main             # instruction nr. 8
done:
  irmovq $0, %rdi      # instruction nr. 9
  halt                 # instruction nr. A
two:
  .quad 0x000000000002
```

(a)

Indicate, for each of the following values in place of D and S, the value of register %rax after program execution.
Assume that the program is running on SEQ hardware.

- 1) D = 1, S = 3
- 2) D = 2, S = 6
- 3) D = 2, S = 3

(b)

Indicate, for each of the following clock cycles during program execution, which program instruction is in which stage of the pipeline (i.e. in which of the F, D, E, M, W pipeline registers). In each field, indicate the instruction number of the instruction there, or blank for pipeline bubble.
Assume D = 1, S = 3, and that the program is running on PIPE hardware (incl. forwarding- and pipeline-control-logic).
Hint: We have pre-filled the stages for the first four cycles.

```
     F   D   E   M   W
1) [0] [ ] [ ] [ ] [ ]
2) [1] [0] [ ] [ ] [ ]
3) [2] [1] [0] [ ] [ ]
4) [3] [2] [1] [0] [ ]
5) [ ] [ ] [ ] [ ] [ ]
6) [ ] [ ] [ ] [ ] [ ]
7) [ ] [ ] [ ] [ ] [ ]
8) [ ] [ ] [ ] [ ] [ ]
9) [ ] [ ] [ ] [ ] [ ]
A) [ ] [ ] [ ] [ ] [ ]
B) [ ] [ ] [ ] [ ] [ ]
C) [ ] [ ] [ ] [ ] [ ]
D) [ ] [ ] [ ] [ ] [ ]
E) [ ] [ ] [ ] [ ] [ ]
F) [ ] [ ] [ ] [ ] [ ]
```

locality (6 points)

Consider `naive_rotate` from **prflab**:

```c
#define RIDX(i,j,n) ((i)*(n)+(j))
void naive_rotate(int dim, pixel *src, pixel *dst)
{
    int i, j;

    for (i = 0; i < dim; i++)
        for (j = 0; j < dim; j++)
            dst[RIDX(dim-1-j, i, dim)] = src[RIDX(i, j, dim)];
}
```

(a)

What kind of locality of data does this program exhibit wrt. the following variables occurring within it?

- `src`
- `dst`

- `i`

For each, choose one of: spatial, temporal, both, or neither.

(b)

What kind of locality of instructions does this program exhibit wrt. the following instructions occurring within it?

- `i=0`
- `j++`
- `inner loop body`

For each, choose one of: spatial, temporal, both, or neither.

caching (6 points)

(a)

Why does thrashing happen?

- High miss penalty
- Cache incoherence
- Cache miss
- Capacity miss

Pick all that apply.

(b)

A cache coherence protocol ensures that...

- ... a read from an address gives you its most up-to-date value.
- ... data in the cache can be accessed in a coherent way.
- ... contents of the cache is consistent with what is contained in memory.

Pick one that applies.

memory (6 points)

(a)

What is a segmentation fault?

- The kernel failed to segment memory into pages in response to a request made by the process.
- The data structures in the heap of the process are no longer aligned in memory.
- The process attempted to access a region of memory which it is not privileged to access.

Pick one that applies.

(b)

What do we gain by having a virtual memory mapped region for shared libraries?

- Backwards compatibility (it's a legacy feature).
- Simplifies access to shared libraries.
- Programs will run faster.
- Programs will consume less memory.

Pick all that apply.

i/o (6 points)

(a)

When a program reads from a file on disk, which of the following events occurs as a result?

- The program executes the `IN` instruction (to input from I/O port).
- The program executes the `INT` (aka. `SYSCALL`) instruction.
- The kernel performs a context switch.

- The disk transfers data to a buffer in the CPU.
- The disk interrupts the CPU when the transfer is complete.

Pick all that apply.

(b)

Why is it important to close file descriptors once you are done using them?

- Not doing so introduces a memory leak.
- Otherwise data may still reside in buffers which haven't been flushed.
- The entity at the other end will not know that the session is complete otherwise.

Pick one that applies.

concurrency (6 points)

Consider the following C source file `t.c`.

C

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

int n = 3;
int k = 0;
int s = 0;

void *incr(void* arg) {
  int tk = ++k;
  int ts = s;
  int i;
  for (i = 0; i < n; i++) {
    ts += tk;
  }
  usleep( ( (float)rand() / RAND_MAX ) * 10 );
  s = ts;
}

int main() {
  pthread_t* tids = malloc( n * sizeof(pthread_t) );
  int i;
  srand(time(NULL));
  for (i = 0; i < n; i++) {
    pthread_create( &(tids[i]), NULL, incr, (void*)NULL );
  }
  for (i = 0; i < n; i++) {
    pthread_join(tids[i], NULL);
  }
  printf("%d\n", s);
}
```

It compiles without error with the following command:

SHELL

```shell
gcc -pthread -o t t.c
```

(a)

Upon executing this program (`./t`), which of the following are possible outputs of the program?

- 30
- 15
- 3
- 9
- 12

Pick all that apply.

(b)

Which of the following variables would need to be protected (e.g. by a `mutex`) to prevent race conditions?

- `n`
- `k`
- `s`

- `tk`
- The `mutex` itself.

Pick all that apply.

miscellaneous (15 points)

(a)

Why does an optimizing C compiler not do code motion on function calls?

- Not all functions are referrentially transparent.
- Statements in a function body cannot substitute a function call expression.
- It is too difficult to detect function calls in source code.

Pick one that applies.

(b)

Suppose process P1 allocates a region on the heap using `malloc`, and gets back an address 0x00000020. Process P1 then sets the contents of the region to all 1s. Suppose that process P2 then attempts to read from 0x00000020. What happens?

- P2 reads in the value 0x00000020 had before P1 overwrote it with 1s (P2 does not have access to P1's address space).
- P2 reads in the the all 1s that P1 wrote to 0x00000020.
- P2 reads in the contents of the physical page that 0x00000020 maps to in its own address space.
- P2 experiences a segmentation fault.

Pick all plausible outcomes.

(c)

On mainstream computer systems, when one process is currently running, how do other processes get to run?

- The running process relinquishes control of the processor core once it is ready to do so.
- An interrupt forces the processor to context-switch to the kernel at regular intervals.
- The kernel process suspends the running process, and decides which process goes next.

Pick one that applies.

(d)

How can a program, running on a core, ensure that an otherwise non-atomic instruction gets executed atomically?

- By pausing execution of all other cores.
- By utilizing a special instruction modifier that locks the memory bus.
- By asking the kernel to perform the instruction on its behalf.

Pick one that applies.

(e)

What is the benefit of the "Everything is a file" idea implemented in Unix systems?

- Can easily back up and restore all kinds of data.
- Facilitates adding, modifying, and deleting, devices on your system.
- A unified API for doing I/O on all kinds of resources.

Pick one that applies.