Recitation 3: Synchronisation

Kai Mast

EL33TH4X0R ?

Which of the following is the best way to wait for two predicates to be true?

(A)

```
with lock:
  while not condA or not condB:
    if not condA:
      condA_cv.wait()
    if not condB:
      condB_cv.wait()
```

(B)

```
with lock:
  while not condA:
      condA_cv.wait()
  while not condB:
      condB_cv.wait()
```

(C)

```
with lock:
  while not condA or not condB:
     condA_cv.wait()
     condB_cv.wait()
```

(D)

```
with lock:
  if not condA or not condB:
    while not condA:
      condA_cv.wait()
    while not condB:
      condB_cv.wait()
```

Which of the following are (virtually) shared by threads within a single process?

[✓ (A) Heap]

(B) Stack

(C) Code / Program Text

[✓ (D) Registers]

Which of the following operations require the executing code to be operating with high privilege?

(A) Implementing a monitor

(B) Performing a semaphore P operation

[✓ (C) Accessing the device registers of an I/O device, e.g. the disk, keyboard, or network card]

[✓ (D) Disabling interrupts]

(E) Making a system call

You are using a semaphore package which provides 3 functions: init(), P(), and V(). Which of the following changes to the package could affect the correctness of your code?

(A) P is modified so that it busy-waits instead of yielding when a resource isn’t available.:

[✓ (B) init is modified so that it only accepts 0 or 1 as an initial value.]

(C) The implementation stores the count in an unsigned int instead of a signed int.

(D) V is modified so that it wakes the thread that most recently called P.

(E) Asserts are removed from all three functions.

What are the two main correctness properties for (operating) systems?

(A) Safety and Soundness

(B) Soundness and Correctness

(C) Freedom and Democracy

[✓ (D) Safety and Liveness]

(E) Concurrency and Performance

Which of the following statements about threads is false?

[✓ (A) Multi-threading is only useful on a multi-core processor.]

(B) Multi-threading is only useful when a task can be parallelized.

(C) There are performance benefits to running threads of the same process one after the other on the same processor.

[✓ (D) Multi-threading requires operating system support for managing multiple PCBs]

Compare and Set:
Use this simple primitive
for the next two questions

```
ATOMIC bool CAS(int *addr, int oldval, int newval)
{
    if (*addr != oldval)
        return false;


    *addr = newval;
    return true;
}
```

Implement Test-And-Set

```
bool TAS(int *addr)
{
        return CAS(addr, 0, 1);
}
```

Implement Atomic Increment

```
void increment(int *addr)
{
        int oldval = *addr;
        while (!CAS(addr, oldval, oldval+1))
                oldval = *addr;
}
```

Implement Atomic List Append

```
struct item {
       // points to previous item added to the list
       // (NULL for first item)
       struct item *prev;
       int value; // contains the value in this entry
};
// points to last item added to the list (null if list is empty)
struct item *list = NULL;
void add(int val)  { // add value to the list
struct item *node = malloc(sizeof(structitem));
node->value = val;
node->prev = list; //replace  these 2 lines
list = node; //with thread safe code
```

Implement Atomic List Append

```
struct item {
       // points to previous item added to the list
       // (null for first item)
       struct item *prev;
       int value; // contains the value in this entry
};
// points to last item added to the list
struct item *list = NULL;
void add(int val)  { // add value to the list
struct item *node = malloc(sizeof(structitem));
node->value = val;
do {
node->prev = list;
} while (!CAS(&list, node->prev, node);
```