# COMP 7500/7506 Lecture 17 Project 3-5 AUbatch – CVs and Threads

**Exercise 1:** Please complete the consumer code using condition variables.

char consumer() {

char c;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

while (count == 0) {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

count--;

c = buffer[tail];

tail++;

if (tail == SIZE) {

tail = 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

return c;

}

**Exercise 2:** How to implement cv\_wait()?

void cv\_wait(struct cv \*cv, struct lock \*lock) {

use assert to check input cv and lock;

turn off interrupts;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

sleep the thread until someone signals cv;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

turn on interrupts to the previous level;

}

**Exercise 3:** How to implement cv\_signal()?

void cv\_signal(struct cv \*cv, struct lock \*lock) {

use assert to check cv and lock;

turn off \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

/\* Question: How to implement the following IF \*/

if (this thread does not hold lock)

panic("cv\_signal error: cv %s at %p, lock %s at

%p.\n", cv->name, cv, lock->name, lock);

/\* see also how to wakeup a thread Slide 15 \*/

wakeup one thread using indicator “cv”;

turn on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

**Exercise 4:** What are the differences between processes and threads? (Hint: sharing vs. non-sharing)

**Exercise 5:** What are the advantages of threads over (multi) processes?

5.1 Why it takes less time to switch between two threads within the same process?

5.2 Why the communication overheads among multiple threads is low?

**Exercise 6:** How to solve the performance bottleneck problem in the departmental tour?

**Exercise 7:** Programming Models: (Hint: Global vs. Private)

7.1 Process based models assume that all data associated with a process is \_\_\_\_\_\_\_\_, by default, unless otherwise specified.

7.2 Lightweight processes and threads assume that all memory is \_\_\_\_\_\_\_.

**Exercise 8:** What is the difference between the following two code segments?

for (row = 0; row < n; row++)

for (column = 0; column < n; column++)

c[row][column] =

dot\_product( get\_row(a, row),

get\_col(b, col));

for (row = 0; row < n; row++)

for (column = 0; column < n; column++)

c[row][column] =

create\_thread( dot\_product(get\_row(a, row), get\_col(b, col)));