Report to

Sujoy Debnath

The City College of New York

Grove School of Engineering

Operating Systems Lab

CSC 33200 H

Lab #6 Report

Submitted By: Yauheni Patapau

Submission Date: 12/19/2021

The purpose of this assignment is to develop two programs (one using processes, another one using threads) and synchronize them. The problem consists of 3 smoker processes/threads and 1 agent process/thread. Each smoker continuously rolls a cigarette from three ingredients: matches paper and tobacco. One of the smokers has paper, another has tobacco, and the third has matches. The other ingredients to roll paper are supplied by the agent. Agent 10 times places two random ingredients on the table, and the smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on the completion.

Program with 4 processes has 5 mutexes. One is used to lock the critical section. Another 3 are used to lock each smoker until two remaining ingredients that the smoker needs are placed on the table. The last mutex is used to lock the agent (make it sleep) until the smoker who finished rolling the paper signals agent to resume distribution. Mutex to lock critical section is set to 1, others are set to 0. First, the agent process is created. Then inside of the agent process three child processes (smokers) are created. Each smoker has a similar structure. They have an infinite while loop to run until the agent processes terminates them using kill.

if (smoker[0] == 0) {

while (true) {

P(smoker\_match);

P(mutex);

int r = rand() % 6 + 1;

printf("\nSmoker with matches is smoking %ds\n", r);

V(agent);

V(mutex);

sleep(r);

}

exit(0);

}

Inside the loop, the smoker semaphore is decreased to prevent it from running in an infinite loop. Mutex is decreased to lock critical section. After Smoker rolls paper semaphore agent is increased to signal agent resume distribution. Mutex increased to unlock the critical section. Sleep is used to smoke some random number of seconds.

While loop for the agent process is at the end after children were declared. Otherwise, there would be an error if children weren’t created before we start running the agent while loop. While loop runs until the counter is 0. It decreases by 1 each time when two ingredients were distributed. Mutex at the top locks the critical section. A random number from 1 to 3 is generated after. The result of this number determines what two ingredients will be distributed and what smoker semaphore will be unlocked. The critical section is unlocked at the end and the agent semaphore is used to sleep until one of the smokers sends a signal. After while loop finishes, kill is used to terminate smokers.

while (counter) { // agent part

P(mutex); // lock critical section

printf("\n%d ---Agent starts distributing ingredients---\n", counter);

int rand\_num = rand() % 3 + 1; // generate random number from 1 to 3 to put two ingredients on table

if (rand\_num == 1) {

printf("Agent places tobacco and paper\n");

V(smoker\_match); // increase semaphore for smoker with matches

} else if (rand\_num == 2) {

printf("Agent places tobacco and matches\n");

V(smoker\_paper); // increase semaphore for smoker with paper

} else {

printf("Agent places paper and matches\n");

V(smoker\_tobacco); // increase semaphore for smoker with tobacco

}

counter--; // decrease counter

V(mutex); // unlock critical section

P(agent); // agent sleeps until one of the smokers sends a signal

}

kill(smoker[0], SIGTERM); // using kill to terminate smoker processes, otherwise they will never terminate because of the infinite while loop

kill(smoker[1], SIGTERM);

kill(smoker[2], SIGTERM);

exit(0);

Program with threads also uses for mutexes. In the same way four of them are locked prior creating threads. Only mutex lock for critical section is not locked initially. Two void functions are used for threads. One is for smoker threads and agent\_runner is for agent thread.

pthread\_mutex\_init(&lock, NULL); // initialize mutexes

pthread\_mutex\_init(&match, NULL);

pthread\_mutex\_init(&paper, NULL);

pthread\_mutex\_init(&tobacco, NULL);

pthread\_mutex\_init(&agent, NULL);

pthread\_mutex\_lock(&match); // lock match, paper and tobacco to stop threads from running before ingredients were distributed

pthread\_mutex\_lock(&paper);

pthread\_mutex\_lock(&tobacco);

pthread\_mutex\_lock(&agent); // lock agent to prevent it running before smoker rolled a cigarette

Smoker threads utilize the runner function based on the passing parameter. The runner function has three conditions for each smoker to use a specific mutex lock. The placement of pthread mutexes is identical to the placement in the program with processes. While loop for agent also runs 10 times and terminates signaling other threads to terminate as well using integer run that is set from initial 1 to 0.

void \*agent\_runner() {

while (counter) { // while loop to run 10 times

pthread\_mutex\_lock(&lock); // locking critical section

printf("\n%d ---Agent starts distributing ingredients---\n", counter);

int rand\_num = rand() % 3 + 1;

if (rand\_num == 1) {

printf("Agent places tobacco and paper\n");

pthread\_mutex\_unlock(&match); // unlocking mutex for particular smoker who can roll cigarette

} else if (rand\_num == 2) {

printf("Agent places tobacco and matches\n");

pthread\_mutex\_unlock(&paper);

} else {

printf("Agent places paper and matches\n");

pthread\_mutex\_unlock(&tobacco);

}

counter--;

pthread\_mutex\_unlock(&lock); // unlocking critical section

pthread\_mutex\_lock(&agent); // here agent will sleep until one the smokers unlocks agent mutex

}

run = 0; // after while loop runs 10 times, set run flag to 0 to terminate smoker threads

pthread\_mutex\_unlock(&match); // unlocking smoker mutexes in case if one of them got stuck on its inner mutex and can't be terminated even when run = 0

pthread\_mutex\_unlock(&paper);

pthread\_mutex\_unlock(&tobacco);

pthread\_exit(0); // exit agent

}

void \*runner(void \*runner\_value) {

int mutex\_index = \*((int \*)runner\_value); // taking passed parameter

while (run) { // while loop that runs until run is set to 1

if (mutex\_index == 1) { // if condition to use speficic mutex lock based on the passed parameter (1 for matches, 2 for paper, 3 for tobacco)

pthread\_mutex\_lock(&match);

if (run == 0) { // this if condition is necessary because thread could get stuck on its smoker mutex and won't be terminated even when run is set to 0

pthread\_exit(0);

}

} else if (mutex\_index == 2) {

pthread\_mutex\_lock(&paper);

if (run == 0) {

pthread\_exit(0);

}

} else if (mutex\_index == 3) {

pthread\_mutex\_lock(&tobacco);

if (run == 0) {

pthread\_exit(0);

}

} else {

(pthread\_exit(0));

}

pthread\_mutex\_lock(&lock); // lock for critical section

int r = rand() % 6 + 1;

if (mutex\_index == 1) { // if condtions to smoke based on ingredient that smokers already have

printf("\nSmoker with matches is smoking %ds\n", r);

}

if (mutex\_index == 2) {

printf("\nSmoker with paper is smoking %ds\n", r);

}

if (mutex\_index == 3) {

printf("\nSmoker with tobacco is smoking %ds\n", r);

}

pthread\_mutex\_unlock(&agent); // unlock agent

pthread\_mutex\_unlock(&lock); // unlock critical section

sleep(r); // smoke random number of seconds

}

pthread\_exit(0);

}