//semaphore signaling mechanism

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

sem\_t semaphore;

void \*process(void \*processId)

{

int id = \*((int \*)processId);

printf("P%d is waiting on the semaphore...\n", id);

sem\_wait(&semaphore);

printf("P%d enters the critical section.\n", id);

printf("P%d is using the resource.\n", id);

int sleep(2);

sem\_post(&semaphore);

printf("P%d releases the semaphore and exits the critical section.\n", id);

return NULL;

}

int main()

{

sem\_init(&semaphore, 0, 4);

pthread\_t threadIds[5];

int processIds[] = {1, 2, 3, 4, 5};

for (int i = 0; i < 5; ++i)

{

pthread\_create(&threadIds[i], NULL, process, &processIds[i]);

}

for (int i = 0; i < 5; ++i)

{

pthread\_join(threadIds[i], NULL);

}

sem\_destroy(&semaphore);

return 0;

}

Output:

