# ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

# TRƯỜNG ĐẠI HỌC BÁCH KHOA


# KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

# BÁO CÁO

## LAB 4 : MULTI-TASKING AND SCHEDULER ACTIVATIONS

**GIẢNG VIÊN HƯỚNG DẪN: BÙI XUÂN GIANG**

**SINH VIÊN THỰC HIỆN: NGUYỄN TẤN PHÁT**

**MSSV: 2352888**

**LỚP: CN01**

**Thành phố Hồ Chí Minh – 2025**

**PROBLEM 1**

**Code:**

```c
#include "bktpool.h"
#include <signal.h>
#include <stdio.h>

#define _GNU_SOURCE
#include <linux/sched.h>
#include <sys/syscall.h>      /* Definition of SYS_* constants */
#include <unistd.h>

//#define DEBUG
#define INFO
#define WORK_THREAD

void * bkwrk_worker(void * arg) {
  sigset_t set;
  int sig;
  int s;
  int i = * ((int * ) arg); // Default arg is integer of workid
  struct bkworker_t * wrk = & worker[i];

  /* Taking the mask for waking up */
  sigemptyset( & set);
  sigaddset( & set, SIGUSR1);
  sigaddset( & set, SIGQUIT);

#ifdef DEBUG
  fprintf(stderr, "worker %i start living tid %d \n", i, getpid());
  fflush(stderr);
#endif

  while (1) {
    /* wait for signal */
    s = sigwait( & set, & sig);
    if (s != 0)
      continue;

#ifdef INFO
    fprintf(stderr, "worker wake %d up\n", i);
#endif

    /* Busy running */
    if (wrk -> func != NULL)
      wrk -> func(wrk -> arg);

    /* Advertise I DONE WORKING */
    wrkid_busy[i] = 0;
    worker[i].func = NULL;
    worker[i].arg = NULL;
    worker[i].bktaskid = -1;
  }
}

```

```
53  int bktask_assign_worker(unsigned int bktaskid, unsigned int wrkid) {
54    if (wrkid < 0 || wrkid > MAX_WORKER)
55      return -1;
56
57    struct bktask_t * tsk = bktask_get_byid(bktaskid);
58
59    if (tsk == NULL)
60      return -1;
61
62    /* Advertise I AM WORKING */
63    wrkid_busy[wrkid] = 1;
64
65    worker[wrkid].func = tsk -> func;
66    worker[wrkid].arg = tsk -> arg;
67    worker[wrkid].bktaskid = bktaskid;
68
69    printf("Assign tsk %d wrk %d \n", tsk -> bktaskid, wrkid);
70    return 0;
71  }
72
73  int bkwrk_create_worker() {
74    unsigned int i;
75
76    for (i = 0; i < MAX_WORKER; i++) {
77  #ifdef WORK_THREAD
78      void ** child_stack = (void ** ) malloc(STACK_SIZE);
79      unsigned int wrkid = i;
80      pthread_t threadid;
81
82      sigset_t set;
83      int s;
84
85      sigemptyset( & set);
86      sigaddset( & set, SIGQUIT);
87      sigaddset( & set, SIGUSR1);
88      sigprocmask(SIG_BLOCK, & set, NULL);
89
90      /* Stack grow down - start at top*/
91      void * stack_top = child_stack + STACK_SIZE;
92
93      wrkid_tid[i] = clone( & bkwrk_worker, stack_top,
94        CLONE_VM | CLONE_FILES,
95        (void * ) & i);
96  #ifdef INFO
97      fprintf(stderr, "bkwrk_create_worker got worker %u\n", wrkid_tid[i]);
98  #endif
99
100     usleep(100);
```

```
101
102       #else
103
104       /* TODO: Implement fork version of create worker */
105   #endif
106
107     }
108
109     return 0;
110   }
111
112   int bkwrk_get_worker() {
113     for (int i = 0; i < MAX_WORKER; i++) {
114       if (wrkid_busy[i] == 0) {
115           return i;
116       }
117     }
118     return -1;
119   }
120
121   int bkwrk_dispatch_worker(unsigned int wrkid) {
122
123   #ifdef WORK_THREAD
124     unsigned int tid = wrkid_tid[wrkid];
125
126     /* Invalid task */
127     if (worker[wrkid].func == NULL)
128       return -1;
129
130   #ifdef DEBUG
131     fprintf(stderr, "brkwrk dispatch wrkid %d - send signal %u \n", wrkid, tid);
132   #endif
133
134     syscall(SYS_tkill, tid, SIG_DISPATCH);
135     #else
136     /* TODO: Implement fork version to signal worker process here */
137
138   #endif
139   }
```

**Output:**

```
mrcopper@MrCopper:/mnt/c/Users/ADMIN/OneDrive - ntpdeveloper/BK Năm 2/Hệ điều hành/LAB 4/lab4-student/p1threadpool$ ./mypool
bkwrk_create_worker got worker 425
bkwrk_create_worker got worker 426
bkwrk_create_worker got worker 427
bkwrk_create_worker got worker 428
bkwrk_create_worker got worker 429
bkwrk_create_worker got worker 430
bkwrk_create_worker got worker 431
bkwrk_create_worker got worker 432
bkwrk_create_worker got worker 433
bkwrk_create_worker got worker 434
Assign tsk 0 wrk 0
Assign tsk 1 wrk 1
Assign tsk 2 wrk 2
worker wake 0 up
Task func - Hello from 1
worker wake 1 up
worker wake 2 up
Task func - Hello from 2
Task func - Hello from 2
Task func - Hello from 5
```

## PROBLEM 2

### Code:

```
101
102     #else
103         pid_t pid = fork();
104         if (pid < 0) {
105             return -1;
106         } else if (pid == 0) {
107             sigset_t set;
108             sigemptyset(&set);
109             sigaddset(&set, SIGUSR1);
110             sigaddset(&set, SIGQUIT);
111             sigprocmask(SIG_BLOCK, &set, NULL);
112
113             while (1) {
114                 int sig;
115                 if (sigwait(&set, &sig) != 0)
116                     continue;
117                 if (worker[i].func != NULL)
118                     worker[i].func(worker[i].arg);
119                 wrkid_busy[i] = 0;
120                 worker[i].func = NULL;
121                 worker[i].arg = NULL;
122                 worker[i].bktaskid = -1;
123             }
124             //_exit(0);
125         } else {
126 #ifdef INFO
127             fprintf(stderr, "bkwrk_create_worker got worker %u\n", pid);
128 #endif
129             wrkid_tid[i] = pid;
130             usleep(100);
131         }
132 #endif
```

### Output:

```
mrcopper@MrCopper:/mnt/c/Users/ADMIN/OneDrive - ntpdeveloper/BK Năm 2/Hệ điều hành/LAB 4/lab4-student/p1threadpool$ ./mypool
bkwrk_create_worker got worker 700
bkwrk_create_worker got worker 701
bkwrk_create_worker got worker 702
bkwrk_create_worker got worker 703
bkwrk_create_worker got worker 704
bkwrk_create_worker got worker 705
bkwrk_create_worker got worker 706
bkwrk_create_worker got worker 707
bkwrk_create_worker got worker 708
bkwrk_create_worker got worker 709
Assign tsk 0 wrk 0
Assign tsk 1 wrk 1
worker wake 0 up
Assign tsk 2 wrk 2
Assign tsk 2 wrk 2
Task func - Hello from 1
worker wake 1 up
worker wake 2 up
Task func - Hello from 2
Task func - Hello from 2
Task func - Hello from 5
```

## PROBLEM 3

### Code:

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    for(int i = 0; i < 5; i++) {
        pid_t pid = fork();
        if(pid == 0) {  // Tiến trình con
            usleep(300000 * i);
            printf("Child %d is running (PID: %d)\n", i, getpid());
            return 0;
        }
    }
    // Join phase
    for(int i = 0; i < 5; i++) {
        wait(NULL);
    }
    printf("All children have finished\n");
    return 0;
}

```

### Output:

```
mrcopper@MrCopper:/mnt/c/Users/ADMIN/OneDrive - ntpdeveloper/BK Năm 2/Hệ điều hành/LAB 4/lab4-student/p3forkjoin$ gcc -o fork_join fork_join.c -lpthread
mrcopper@MrCopper:/mnt/c/Users/ADMIN/OneDrive - ntpdeveloper/BK Năm 2/Hệ điều hành/LAB 4/lab4-student/p3forkjoin$ ./fork_join
Child 0 is running (PID: 998)
Child 1 is running (PID: 999)
Child 2 is running (PID: 1000)
Child 3 is running (PID: 1001)
Child 4 is running (PID: 1002)
All children have finished
```