**강의명: 유닉스 시스템**

**실습 번호: 9**

**실습 제목: 쓰레드(Threads)**

**학생 이름: 황귀훈**

**학번: 201710885**

**1. 쓰레드 생성**

**1.1**

번호 1 쓰레드 생성은 프로세스 내에서 새로운 thread를 생성하여 수행시키는 프로그램이다. 함수 print\_ids()는 getpid(), pthread\_self()함수를 이용하여 현재 쓰레드의 pid와 tid를 출력하는 함수이다. 함수 pthread\_create()는 인수에 주어지는 start\_routine및 arg를 가지고 새로운 thread를 수행시킨다. 새로 생성된 thread의 id는 첫째 인수로 받는다. 이 때 인수 attr에는 default값인 NULL을 준다. 만약 이 함수의 반환 값이 0이 아니라면 에러 메시지를 출력한다.

pthread\_create()의 인수로 주어지는 start\_routine은 thread\_function()으로 “new thread”를 출력하고 return한다. 이 때 start routine은 새로운 thread를 생성할 때 그 thread의 수행 시작 지점이다.

따라서 이 프로그램을 실행시키면 main thread에서 new thread를 생성하고 각각 thread의 pid와 tid를 출력하고 종료한다.

**1.2**

****

명령 줄 “./thread”의 수행 결과는 위와 같다. main함수에서 print\_ids()함수의 인수로 "main thread"를 주어 printf("%s: pid=%u, tid=%x\n", s, (unsigned int) pid, (unsigned int) tid)의 결과로 첫 번째 줄과같이 main thread의 pid와 tid를 출력한다. 그 다음 새로 생성된 thread는 start\_routine인 thread\_function()에서 print\_ids()의 인수로 “new thread”를 주어 호출하기 때문에 printf("%s: pid=%u, tid=%x\n", s, (unsigned int) pid, (unsigned int) tid)의 결과로 새로 생긴 new thread의 pid와 tid가 출력되는 것이다.

**2. 쓰레드 종료**

**2.1**

번호 2 쓰레드 종료는 두 개의 쓰레드를 생성한 뒤 pthread\_join()함수를 호출하여 생성된 두 개의 쓰레드를 terminate시키는 프로그램이다. 이 때 각 쓰레드의 start routine에서 반환 값을 다르게 하여 pthread\_join()를 호출한 뒤 그 반환 값을 출력하여 확인해본다.

**2.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

명령 줄 “./terminate”실행 결과가 위와 같은 이유는 다음과 같다. thread\_function1()은 ("thread\_function1 exit: %d\n", ret)을 출력(ret==1)하고 1을 반환한다. thread\_function2()은 ("thread\_function2 exit: %d\n", ret )을 출력(ret==2)하고 2를 반환한다. Main()함수를 보면 pthread\_create()함수를 통해 두개의 쓰레드를 생성함을 알 수 있다. 이 때 “thread\_function1 exit: 1”, “thread\_function2 exit: 2”가 출력된다. (start routine에 의해) 그 후 pthread\_join()함수를 호출하여 인수로 주어진 thread가 종료될 때까지 block되고, 그 thread가 종료되면 ("thread\_function1 join: %d\n", \*ret1)을 출력한다. (이 때의 ret1은 thread\_function1()의 반환 값) 그 후 pthread\_join()에 인수로 나머지 thread를 주어 호출하면 그 쓰레드가 종료되길 기다리다, 종료가 되면 ("thread\_function2 join: %d\n", \*ret2)을 출력한다. (이 때의 ret2는 thread\_function2()의 반환 값) 그 후 프로그램은 종료된다.

따라서 명령 줄 “./terminate”실행 결과 위의 사진처럼 출력된다.

**3. 쓰레드 공유 변수**

**3.1**

번호 3 쓰레드 공유 변수는 앞 실습 번호 2의 프로그램에 있는 2개의 thread 시작 함수를 제거하고 대신 공유 변수 i를 더하고 빼기를 이중 for문을 사용하여 10000x10000회 반복하는 시작함수를 추가한 뒤 프로그램이 종료되기 전 그 공유 변수 i의 값을 출력해 보는 프로그램이다.

이 때 두 쓰레드가 공유 변수를 read/write하는데 변수 값이 의도대로 보장되지 않을 수도 있다. 이번 프로그램에서 두 쓰레드의 시작함수는 같은 회 동안 i를 1더하고 1빼기 때문에 의도한 바로는 프로그램이 종료되고 출력된 공유 변수 i의 값은 0이어야 한다.

**3.2**

**텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명**

명령 줄 “./share”를 4회 실행한 결과이다. 출력된 공유 변수 i의 값을 보면 매 수행 마다 값이 다름을 알 수 있다. 프로그램을 작성하며 기대한 공유변수 i의 값은 0이었지만 0도 아닐 뿐 더러 그 값들이 매 수행마다 달랐는데 이유는 다음과 같다. 두 쓰레드가 공유 변수 i를 read할 때 0이 아닐 때 read를 해서 1을 더하고 1을 빼도 0이 되지 않는다. 이와 같이 두 쓰레드가 Shared변수를 read/write하는 전후에 동기화를 해주지 않는다면 그 값이 일정하지 않고 항상 바뀔 것이다.

race condition에 의해 두 쓰레드의 생성(exit출력)과 종료(join출력)의 순서는 예측할 수 없다. 같은 쓰레드 내에선 exit이 출력되고 그 다음 join이 출력된다.

**4. 쓰레드 동기화**

**4.1**

**#include <unistd.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <pthread.h>**

**int shared\_i = 0;**

**pthread\_mutex\_t lock;**

**void \*thread\_function1(void \*arg)**

**{**

**static int ret = 1;**

**int i, j;**

**for (i = 0; i < 10000; i++) {**

**for (j = 0; j < 10000; j++) {**

**pthread\_mutex\_lock(&lock);**

**shared\_i++;**

**shared\_i--;**

**pthread\_mutex\_unlock(&lock);**

**}**

**}**

**printf("thread\_function1 exit: %d\n", ret);**

**pthread\_exit((void \*) &ret);**

**}**

**void \*thread\_function2(void \*arg)**

**{**

**static int ret = 2;**

**int i, j;**

**for (i = 0; i < 10000; i++) {**

**for (j = 0; j < 10000; j++) {**

**pthread\_mutex\_lock(&lock);**

**shared\_i++;**

**shared\_i--;**

**pthread\_mutex\_unlock(&lock);**

**}**

**}**

**printf("thread\_function2 exit: %d\n", ret);**

**pthread\_exit((void \*) &ret);**

**}**

**int main(void)**

**{**

**int err;**

**pthread\_t tid1, tid2;**

**int \*ret1, \*ret2;**

**err = pthread\_create(&tid1, NULL, thread\_function1, NULL);**

**if (err != 0)**

**exit(1);**

**err = pthread\_create(&tid2, NULL, thread\_function2, NULL);**

**if (err != 0)**

**exit(1);**

**err = pthread\_join(tid1, (void \*) &ret1);**

**if (err != 0)**

**exit(1);**

**printf("thread\_function1 join: %d\n", \*ret1);**

**err = pthread\_join(tid2, (void \*) &ret2);**

**if (err != 0)**

**exit(1);**

**printf("thread\_function2 join: %d\n", \*ret2);**

**printf("shared\_i = %d\n", shared\_i);**

**exit(0);**

**}**

번호 4 쓰레드 동기화는 앞 실습 번호 3의 프로그램을 수정하여 공유 변수 수 shared\_i의 최종 값이 항상 0이 출력 되도록 해야 한다. 따라서 공유 변수를 read/write하는 부분 전후에 pthread\_mutex\_lock 및 pthread\_mutex\_unlock를 불러서 두 쓰레드를 synchronization 해야 한다.

두 쓰레드를 동기화 해주면 쓰레드1이 공유변수 i를 불러와 10000x10000회 동안 한 회에 i에 1을 더하고 1을 빼는 수행이 끝나고, 0을 저장하면 그 때 쓰레드2가 공유변수 i(=0)를 불러와 똑같이 10000x10000회 동안 회당 i를 1더하고 1빼는 수행을 반복한 뒤 저장하여 최종 공유 변수 i가 0이 된다.

**4.2**

**텍스트, 신문, 영수증이(가) 표시된 사진

자동 생성된 설명**

따라서 두 쓰레드가 공유변수 i를 사용하는 부분을 pthread\_mutex\_lock 및 pthread\_mutex\_unlock로 감싸서 동기화를 했기 때문에 명령 줄 “./synchro”를 여러 번 수행하더라도 공유 변수 i는 0을 출력한다. race condition에 의해 두 쓰레드의 생성(exit출력)과 종료(join출력)의 순서는 예측할 수 없다. 같은 쓰레드 내에선 exit이 출력되고 그 다음 join이 출력된다.