

Міністерство освіти і науки України
НТУУ «КПІ ім. Ігоря Сікорського»
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №1
з дисципліни «Технології паралельних обчислень в
енергетичних комплексах»
Тема «Оброблення сигналів в багатопотокових застосунках»
Варіант №19

Студента 3-го курсу НН ІАТЕ гр. ТР-12

Ковальова Олександра

Перевірив: ас., Софієнко А. Ю.

Мета роботи. Опанувати техніку оброблення сигналів у багатопотоковому середовищі.

Хід роботи

1) Пересвідчитись у тому, що потоки в ОС Linux реалізовані як група процесів, яку очолює головний потік. Функція getpgrp.

Програмний код:

```
#include <stdio.h>
#include <sys/syscall.h>
#include <pthread.h>
#include <unistd.h>

#define NUM_THREADS 5

// Function executed by every thread
void *threadFunction(void *threadID) {
    long tid = (long) threadID;
    pid_t local_pid = syscall(SYS_gettid);
    printf("Thread number: %ld | PID: %d | PID_SELF: %lu | LWP_ID: %ld | PGID: %d\n",
           tid, getpid(), pthread_self(), (long) local_pid, getpgrp());
    pthread_exit(NULL);
}

int main() {
    // Threads ids
    pthread_t threads[NUM_THREADS];

    // Creating threads
    for (long i = 0; i < NUM_THREADS; i++) {
        if (pthread_create(&threads[i], NULL, threadFunction, (void *)i) != 0) {
            fprintf(stderr, "Error creating thread %ld\n", i);
            return 1;
        }
    }

    while(1) {
        sleep(1);
    }

    // Waiting for each thread to complete
    for (long i = 0; i < NUM_THREADS; i++) {
        if (pthread_join(threads[i], NULL) != 0) {
            fprintf(stderr, "Error joining thread %ld\n", i);
            return 1;
        }
    }

    return 0;
}
```

Результати роботи програми:

```
xairaven@host:~/PCT/Lab1$ gcc task1.c
xairaven@host:~/PCT/Lab1$ ./a.out
Thread number: 4 | PID: 1932 | PID_SELF: 139844365682240 | LWP_ID: 1937 | PGID: 1932
Thread number: 3 | PID: 1932 | PID_SELF: 139844374074944 | LWP_ID: 1936 | PGID: 1932
Thread number: 2 | PID: 1932 | PID_SELF: 139844382467648 | LWP_ID: 1935 | PGID: 1932
Thread number: 1 | PID: 1932 | PID_SELF: 139844390860352 | LWP_ID: 1934 | PGID: 1932
Thread number: 0 | PID: 1932 | PID_SELF: 139844399253056 | LWP_ID: 1933 | PGID: 1932
^C
xairaven@host:~/PCT/Lab1$
```

1817	root	20	0	234M	8792	7484	S	0.0	0.4	0:00.08	/usr/libexec/upowerd
1819	root	20	0	234M	8792	7484	S	0.0	0.4	0:00.00	/usr/libexec/upowerd
1820	root	20	0	234M	8792	7484	S	0.0	0.4	0:00.00	/usr/libexec/upowerd
1939	xairaven	20	0	106M	1632	1504	S	0.0	0.1	0:00.00	./a.out
1945	root	20	0	16924	10748	8592	S	0.0	0.5	0:00.01	sshd: xairaven [priv]
1947	root	20	0	16924	10860	8704	S	0.0	0.5	0:00.00	sshd: xairaven [priv]
2101	xairaven	20	0	17440	8388	5876	S	0.7	0.4	0:00.04	sshd: xairaven@pts/0

2) Напишіть багатопотокову програму та з'ясуйте, якому потоку буде доставлений сигнал, якщо декілька потоків одночасно очікують надходження того самого сигналу за допомогою функції `sigwait`.

```
#include <stdio.h>
#include <pthread.h>
#include <signal.h>
#include <unistd.h>
#include <sys/syscall.h>

#define NUM_THREADS 5

// Array for storing thread ids
pthread_t threads[NUM_THREADS];

// Callback for all threads
void *threadFunction(void *threadID) {
    long tid = (long)threadID;

    printf("Thread %ld is waiting for SIGINT...\n", tid);

    // Set for sigwait
    sigset_t set;
    sigemptyset(&set);
    sigaddset(&set, SIGINT);

    // Waiting for SIGINT
    int received_signal;
    sigwait(&set, &received_signal);

    pid_t local_pid = syscall(SYS_gettid);
    printf("Thread %ld (PID %ld) received the signal: %d\n", tid, (long) local_pid, received_signal);

    pthread_exit(NULL);
}

int main() {
    // Blocking SIGINT for the main thread
    sigset_t main_set;
    sigemptyset(&main_set);
    sigaddset(&main_set, SIGINT);
    pthread_sigmask(SIG_BLOCK, &main_set, NULL);

    // Creating threads
    for (long i = 0; i < NUM_THREADS; i++) {
        if (pthread_create(&threads[i], NULL, threadFunction, (void *)i) != 0) {
            fprintf(stderr, "Error creating thread %ld\n", i);
            return 1;
        }
    }

    // Waiting for SIGINT in the main thread
    printf("Main thread is waiting for SIGINT...\n");
    int received_signal;
    sigwait(&main_set, &received_signal);


    printf("Main thread received the signal: %d\n", received_signal);

    // Exiting from all threads
    for (long i = 0; i < NUM_THREADS; i++) {
        if (pthread_join(threads[i], NULL) != 0) {
            fprintf(stderr, "Error joining thread %ld\n", i);
            return 1;
        }
    }

    return 0;
}
```

Результати роботи програми:

```
2104 xairaven 20 0 7772 5424 4352 S 0.0 0.3 0:00.00 /usr/lib/openssh/sftp-server
2168 xairaven 20 0 43756 968 880 S 0.0 0.0 0:00.00 ./a.out
2169 xairaven 20 0 43756 968 880 S 0.0 0.0 0:00.00 ./a.out
2170 xairaven 20 0 43756 968 880 S 0.0 0.0 0:00.00 ./a.out
2171 xairaven 20 0 43756 968 880 S 0.0 0.0 0:00.00 ./a.out
2172 xairaven 20 0 43756 968 880 S 0.0 0.0 0:00.00 ./a.out
2173 xairaven 20 0 43756 968 880 S 0.0 0.0 0:00.00 ./a.out
2174 xairaven 20 0 8372 4496 3444 R 0.7 0.2 0:00.10 htop
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit
```



```
xairaven@host:~/PCT/Lab1$ gcc task2.c
xairaven@host:~/PCT/Lab1$ ./a.out
Main thread is waiting for SIGINT...
Thread 4 is waiting for SIGINT...
Thread 3 is waiting for SIGINT...
Thread 2 is waiting for SIGINT...
Thread 1 is waiting for SIGINT...
Thread 0 is waiting for SIGINT...
^CMain thread received the signal: 2
^CThread 0 (PID 2169) received the signal: 2
^CThread 1 (PID 2170) received the signal: 2
^CThread 2 (PID 2171) received the signal: 2
^CThread 3 (PID 2172) received the signal: 2
^CThread 4 (PID 2173) received the signal: 2
xairaven@host:~/PCT/Lab1$ ^C
```

3) Напишіть багатопотокову програму та з'ясуйте, в який спосіб буде оброблений доставлений сигнал у випадку одночасного використання деяким потоком функцій `sigwait` та `sigaction`.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <sys/syscall.h>
#include <unistd.h>

void sigaction_handler(int signo) {
    printf("Received SIGINT using sigaction in thread %ld\n", (long) syscall(SYS_gettid));
    pthread_exit(NULL);
}

void *sigaction_thread(void *arg) {
    // Initializing sigaction handler
    struct sigaction sa;
    sa.sa_handler = sigaction_handler;
    sigaction(SIGINT, &sa, NULL);

    // Blocking thread while waiting for a signal
    pause();

    printf("Exiting sigaction_thread\n");

    return NULL;
}
```

```

void *sigwait_thread(void *arg) {
    sigset_t set;
    int sig;

    // Initializing set of signals that threads are waiting
    sigemptyset(&set);
    sigaddset(&set, SIGINT);

    // Waiting for this signal
    sigwait(&set, &sig);

    printf("Received SIGINT using sigwait in thread %ld\n", (long) syscall(SYS_gettid));

    pthread_exit(NULL);

    return NULL;
}

int main() {
    pthread_t sigaction_thread_id, sigwait_thread_id;

    printf("Waiting for SIGINT (Ctrl+C):\n");
    fflush(stdout);

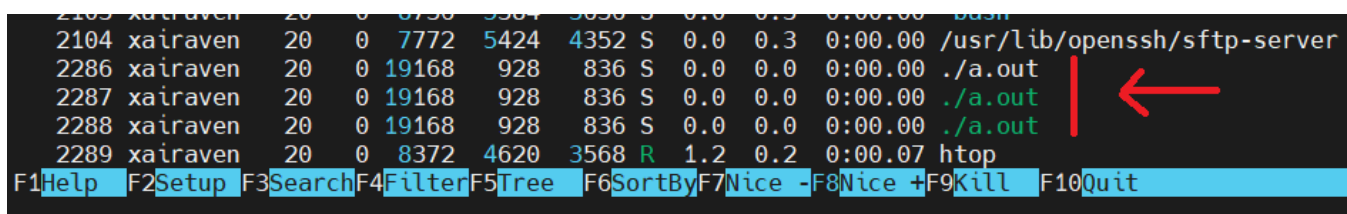
    // Creating threads
    pthread_create(&sigaction_thread_id, NULL, sigaction_thread, NULL);
    pthread_create(&sigwait_thread_id, NULL, sigwait_thread, NULL);

    // Waiting until threads will be done
    pthread_join(sigaction_thread_id, NULL);
    pthread_join(sigwait_thread_id, NULL);

    return 0;
}

```

Результат роботи програми:



```

2103 xairaven 20 0 8738 3584 3588 S 0.0 0.3 0:00.00 bash
2104 xairaven 20 0 7772 5424 4352 S 0.0 0.3 0:00.00 /usr/lib/openssh/sftp-server
2286 xairaven 20 0 19168 928 836 S 0.0 0.0 0:00.00 ./a.out
2287 xairaven 20 0 19168 928 836 S 0.0 0.0 0:00.00 ./a.out
2288 xairaven 20 0 19168 928 836 S 0.0 0.0 0:00.00 ./a.out
2289 xairaven 20 0 8372 4620 3568 R 1.2 0.2 0:00.07 htop
F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 SortBy F7 Nice - F8 Nice + F9 Kill F10 Quit

```

```

xairaven@host:~/PCT/Lab1$ gcc ./task3.c
xairaven@host:~/PCT/Lab1$ ./a.out
Waiting for SIGINT (Ctrl+C):
^CReceived SIGINT using sigaction in thread 2286
^CReceived SIGINT using sigaction in thread 2287
^CReceived SIGINT using sigwait in thread 2288
xairaven@host:~/PCT/Lab1$

```

Контрольні запитання:

1) Чим є сигнали та з якою метою їх використовують в ОС UNIX?

Сигнали в ОС UNIX є способом передачі коротших повідомлень або подій між процесами. Вони використовуються для сповіщення про події, такі як помилки, завершення виконання процесу чи інші зміни у стані програми. Сигнали грають важливу роль у взаємодії між процесами та дозволяють управляти їхнім виконанням, реагуючи на різні ситуації.

2) Які види сигналів існують в ОС UNIX та яким чином можна на них реагувати в разі надходження?

В ОС UNIX існує кілька видів сигналів, таких як SIGTERM для закриття процесу, SIGKILL для негайного завершення, SIGINT для переривання введеного з клавіатури та інші. Кожен сигнал має своє призначення та може бути використаний для сповіщення про різні події. В разі надходження сигналу процес може реагувати різними способами. Наприклад, в програмному коді можна встановити обробник сигналу, який виконає певний код або процедуру при надходженні сигналу. Також можливо змінити дію за замовчуванням для певного сигналу, визначивши новий спосіб обробки за допомогою команд або системних викликів. Це надає розробникам можливість ефективного управління поведінкою їхніх програм під час різних обставин.

3) Окресліть основні принципи розроблення оброблювачів сигналів.

Основні принципи розроблення оброблювачів сигналів включають у себе визначення конкретних дій, які повинні виконуватися під час обробки певного сигналу. Розробник повинен уникати використання складного або часомісткого коду в обробнику сигналу, оскільки це може призвести до непередбачуваного поведінки програми. Обробники сигналів слід розробляти так, щоб вони були максимально ефективними та не викликали нових ситуацій, які можуть виникнути через невірне керування сигналами. Важливо також враховувати можливість виникнення сигналів в будь-якому місці виконання програми та враховувати потенційні проблеми з різними операційними системами, оскільки обробники сигналів можуть мати різні особливості залежно від платформи.

4) Як можна реалізувати синхронне оброблення асинхронних сигналів у багатопотоковій програмі?

У багатопотокових програмах синхронне оброблення асинхронних сигналів можна реалізувати за допомогою функцій `sigwait` та `sigaction`. Функція `sigwait` дозволяє спростити оброблення сигналів у синхронному стилі. Замість того, щоб встановлювати обробники сигналів, можна використовувати `sigwait` для блокування потоку до моменту надходження конкретного сигналу. Це дозволяє обробляти сигнали асинхронно, але в синхронному стилі. Крім того, функція `sigaction` надає більше гнучкості в обробці сигналів, оскільки вона дозволяє визначити обробники сигналів, а також вказати флаги та параметри обробки. Це особливо важливо в багатопотокових програмах, де контроль над тим, як обробляються сигнали, має велике значення для уникнення конфліктів та забезпечення правильної синхронізації дій між потоками.