***Лабораторна робота №20***

***«Використання сигналів у ОС Linux програмним шляхом мовою C»***

***Теоретичні відомості***

Сигнали не можуть безпосередньо переносити інформацію, що обмежує їх придатність в якості загального механізму взаємодії між процесами. Проте, кожному типу сигналів присвоєно мнемонічне ім'я (наприклад, ***SIGINT***), яке вказує, для чого зазвичай використовується сигнал цього типу. Імена сигналів визначені в стандартному заголовному файлі ***<signal.h>*** за допомогою директиви препроцесора ***#define***.

Як і слід було очікувати, ці імена відповідають невеликим позитивним цілим числам. З точки зору користувача отримання процесом сигналу виглядає як виникнення переривання. Процес перериває виконання, і управління передається функції-обробникe сигналу. Після закінчення обробки сигналу процес може відновити регулярне виконання. Типи сигналів прийнято задавати спеціальними символьними константами. Системний виклик ***kill()*** призначений для передачі сигналу одного або декількох специфікованим процесам в рамках повноважень користувача.

***#include <sys/types.h>***

***#include <signal.h>***

***int kill(pid\_t pid, int signal);***

Послати сигнал (не маючи повноважень суперкористувача) можна тільки процесу, у якого ефективний ідентифікатор користувача збігається з ефективним ідентифікатором користувача для процесу, що посилає сигнал. Аргумент ***pid*** вказує процес, якому посилається сигнал, а аргумент ***signal*** - який сигнал посилається. Залежно від значення аргументів:

* ***pid > 0*** сигнал посилається процесу з ідентифікатором pid;
* ***pid = 0*** сигнал посилається всім процесам в групі, до якої належить посилає процес;
* ***pid = -1*** і посилаючий процес не є процесом суперкористувача, то сигнал посилається всім процесам в системі, для яких ім'я користувача збігається з ефективним ідентифікатором користувача процесу, що посилає сигнал;
* ***pid = -1*** і посилаючий процес є процесом суперкористувача, то сигнал посилається всім процесам в системі, за винятком системних процесів (зазвичай всім, крім процесів з ***pid = 0*** і ***pid = 1***);
* ***pid < 0***, але не ***-1***, то сигнал посилається всім процесам з групи, ідентифікатор якої дорівнює абсолютному значенню аргументу ***pid*** (якщо дозволяють привілеї);
* якщо ***sig = 0***, то проводиться перевірка на помилку, а сигнал не посилається. Це можна використовувати для перевірки правильності аргументу pid (чи є в системі процес або група процесів з відповідним ідентифікатором).

Системні виклики для установки власного обробника сигналів:

***#include <signal.h>***

***void (\*signal (int sig, void (\*handler) (int)))(int);***

***int sigaction(int sig, const struct sigaction \*act, struct sigaction \*oldact);***

Структура ***sigaction*** має такий вигляд:

***struct sigaction {***

***void (\*sa\_handler)(int);***

***void (\*sa\_sigaction)(int, siginfo\_t \*, void \*);***

***sigset\_t sa\_mask;***

***int sa\_flags;***

***void (\*sa\_restorer)(void);***

***}***

Системний виклик ***signal*** служить для зміни реакції процесу на будь-якої сигнал. Параметр ***sig*** - це номер сигналу, обробку якого належить змінити. Параметр ***handler*** описує новий спосіб обробки сигналу - це може бути вказівник на призначену для користувача функцію-обробник сигналу, спеціальне значення ***SIG\_DFL*** (відновити реакцію процесу на сигнал ***sig*** за замовчуванням) або спеціальне значення ***SIG\_IGN*** (ігнорувати надійшов сигнал ***sig***). Системний виклик повертає покажчик на старий спосіб обробки сигналу, значення якого можна використовувати для відновлення старого способу в разі потреби.

Приклад користувальницької обробки сигналу ***SIGUSR1***.

***void \*my\_handler(int nsig) {***

***//код функції-обробника сигналу***

***}***

***int main()***

***{***

***(void) signal(SIGUSR1, my\_handler);***

***}***

Системний виклик ***sigaction*** використовується для зміни дій процесу при отриманні відповідного сигналу. Параметр ***sig*** задає номер сигналу і може дорівнювати будь-якого номеру. Якщо параметр ***act*** не дорівнює нулю, то нове дію, не пов'язані з сигналом ***sig***, встановлюється відповідно ***act***. Якщо ***oldact*** не дорівнює нулю, то попередня дія записується в ***oldact***.

Більшість типів сигналів **UNIX** призначені для використання ядром, хоча є кілька сигналів, які надсилаються від процесу до процесу:

***SIGALRM*** - сигнал таймера (**alarm clock**). Надсилається процесу ядром при спрацьовуванні таймера. Кожен процес може встановлювати не менше трьох таймерів. Перший з них вимірює минуле реальний час. Цей таймер встановлюється самим процесом за допомогою системного виклику ***alarm()***;

***SIGCHLD*** - сигнал зупинки або завершення дочірнього процесу (**child process terminated or stopped**). Якщо дочірній процес зупиняється або завершується, то ядро ​​повідомить про це батьківському процесу, пославши йому даний сигнал. За замовчуванням батьківський процес ігнорує цей сигнал, тому, якщо в батьківському процесі необхідно отримувати відомості про завершення дочірніх процесів, то потрібно перехоплювати цей сигнал;

***SIGHUP*** - сигнал звільнення лінії (**hangup signal**). Надсилається ядром всім процесам, підключеним до керуючого терміналу (**control terminal**) при відключенні терміналу. Він також посилається всім членам сеансу, якщо завершує роботу лідер сеансу (зазвичай процес командного інтерпретатора), пов'язаного з керуючим терміналом;

***SIGINT*** - сигнал переривання програми (**interrupt**). Надсилається ядром всіх процесів сеансу, пов'язаного з терміналом, коли користувач натискає кнопку переривання. Це також звичайний спосіб зупинки виконується програми;

***SIGKILL*** - сигнал знищення процесу (**kill**). Це досить специфічний сигнал, який посилається від одного процесу до іншого і призводить до негайного припинення роботи отримує сигнал процесу;

***SIGPIPE*** - сигнал про спробу записи в канал або сокет, для яких приймає процес вже завершив;

***SIGPOLL*** - сигнал про виникнення одного з опитуваних подій (**pollable event**). Цей сигнал генерується ядром, коли деякий відкритий дескриптор файлу стає готовим для введення або виведення;

***SIGPROF*** - сигнал профілюючого таймера (**profiling time expired**). Як було згадано для сигналу ***SIGALRM***, будь-який процес може встановити не менше трьох таймерів. Другий з цих таймерів може використовуватися для вимірювання часу виконання процесу в призначеному для користувача і системному режимах. Цей сигнал генерується, коли закінчується час, встановлений у цьому таймері, і тому може бути використаний засобом профілювання програми;

***SIGQUIT*** - сигнал про вихід (**quit**). Дуже схожий на сигнал ***SIGINT***, цей сигнал посилається ядром, коли користувач натискає кнопку виходу використовуваного терміналу. На відміну від ***SIGINT***, цей сигнал призводить до аварійного завершення і скидання образу пам'яті;

***SIGSTOP*** - сигнал зупинки (**stop executing**). Це сигнал управління завданнями, який зупиняє процес. Його, як і сигнал ***SIGKILL***, не можна проігнорувати або перехопити;

***SIGTERM*** - програмний сигнал завершення (**software termination signal**). Програміст може використовувати цей сигнал для того, щоб дати процесу час для «наведення порядку», перш ніж посилати йому сигнал ***SIGKILL***;

***SIGTRAP*** - сигнал трасування переривання (trace trap). Це особливий сигнал, який в поєднанні з системним викликом ptrace використовується відлагоджувачами, такими як **sdb**, **adb**, **gdb**;

***SIGTSTP*** - термінальний сигнал зупинки (**terminal stop signal**). Він формується при натисканні спеціальної клавіші зупинки;

***SIGTTIN*** - сигнал про спробу введення з терміналу фоновим процесом (**background process attempting read**). Якщо процес виконується у фоновому режимі і намагається виконати читання з керуючого терміналу, то йому надсилається цей сигнал. Дія сигналу за замовчуванням - зупинка процесу;

***SIGTTOU*** - сигнал про спробу виведення на термінал фоновим процесом (**background process attempting write**). Аналогічний сигналу ***SIGTTIN***, але генерується, якщо фоновий процес намагається виконати запис в керуючий термінал. Дія сигналу за замовчуванням - зупинка процесу;

***SIGURG*** - сигнал про надходження в буфер сокета термінових даних (**high bandwidth data is available at a socket**). Він повідомляє процесу, що з мережевого з'єднання отримані термінові позачергові дані;

***SIGUSR1*** і ***SIGUSR2*** - призначені для користувача сигнали (**user defined signals 1 and 2**). Так само, як і сигнал ***SIGTERM***, ці сигнали ніколи не надсилаються ядром і можуть використовуватися для будь-яких цілей за вибором користувача;

***SIGVTALRM*** - сигнал віртуального таймера (virtual timer expired). Третій таймер можна встановити так, щоб він вимірював час, яке процес виконує в призначеному для користувача режимі.

Набори сигналів визначаються за допомогою типу ***sigset\_t***, який визначений в заголовки ***<signal.h>***. Вибрати певні сигнали можна, почавши або з повного набору сигналів і видаливши непотрібні сигнали, або з порожнього набору, включивши в нього потрібні. Ініціалізація порожнього і повного набору сигналів виконується за допомогою процедур ***sigemptyset*** і ***sigfillset*** відповідно. Після ініціалізації з наборами сигналів можна оперувати за допомогою процедур ***sigaddset*** і ***sigdelset***, відповідно додають і видаляють зазначені вами сигнали.

Опис даних процедур:

***#include <signal.h>***

***/\* Ініціалізація \*/***

***int sigemptyset (sigset\_t \*set);***

***int sigfillset (sigset\_t \*set);***

***/\* Додавання та видалення сигналів \*/***

***int sigaddset sigset\_t \*set, int signo);***

***int sigdelset (sigset\_t \*set, int signo);***

Процедури ***sigemptyset*** і ***sigfillset*** мають єдиний параметр - покажчик на змінну типу ***sigset\_t***. Виклик sigemptyset инициализирует набір set, виключивши з нього всі сигнали. І, навпаки, виклик ***sigfillset*** инициализирует набір, на який вказує ***set***, включивши в нього всі сигнали. Додатки повинні викликати ***sigemptyset*** або ***sigfillset*** хоча б один раз для кожної змінної типу ***sigset\_t***.

Процедури ***sigaddset*** і ***sigdelset*** приймають в якості параметрів покажчик на ініціалізований набір сигналів і номер сигналу, який повинен бути доданий або вилучений. Другий параметр, ***signo***, може бути символічним ім'ям константи, таким як ***SIGINT***, або справжнім номером сигналу, але в останньому випадку програма виявиться системно-залежною.

***Завдання***

Організувати функціонування процесів наступної структури:

Процеси визначають свою роботу висновком повідомлень виду:

***N pid ppid поточний час (мсек)*** (N - поточний номер повідомлення) на екран. "Батько" одночасно, посилає сигнал ***SIGUSR1*** "синам". "Сини" отримавши даний сигнал, посилають у відповідь "Батькові" сигнал ***SIGUSR2***. "Батько" отримавши сигнал ***SIGUSR2***, через час t = 100 мсек одночасно, посилає сигнал ***SIGUSR1*** "синам". І так далі ... Написати функції-обробники сигналів, які при отриманні сигналу виводять повідомлення про отримання сигналу на екран. При отриманні/посилці сигналу вони виводять відповідне повідомлення:

***N pid ppid поточний час (мсек) син такий-то get / put SIGUSRm***.

Передбачити механізм для визначення "Батьком", від кого з "Синів" був прийнятий сигнал.

***Контрольні питання***

1. Як програмним чином відправити сигнал певному процесу?
2. Які набори сигналів визначені у стандарті POSIX?
3. Які можливості для управління сигналами надає системний виклик ***sigaction***?
4. Які сигнали призначені для дій користувача?
5. Які сигнали у Linux не можуть бути проігноровані?