Синхронізація в ОС UNIX за допомогою сигналів.

1. Загальні відомості

UNIX розглядає процеси як віртуальні процесори, що можуть опрацьовувати переривання. Переривання в розумінні процесів є сигнали. UNIX має два системних виклики: signal() для призначення функції обробника сигналу та kill() для посилання сигналу процесу. Наступна таблиця показує стандартні номери сигналів:

Сигнал	Значення	Коментар
SIGHUP	1	Hangup detected
SIGINT	2	Interrupt from keyboard
SIGQUIT	3	Quit from keyboard
SIGILL	4	Illegal Instruction
SIGTRAP	5	Trace/breakpoint trap
SIGABRT	6	Abort
SIGUNUSED	7	Unused signal
SIGFPE	8	Floating point exeption
SIGKILL	9	Termination signal
SIGUSR1	10	User-defined signal 1
SIGSEGV	11	Invalid memory reference
SIGUSR2	12	User-defined signal 2
SIGPIPE	13	Write to pipe with no readers
SIGALRM	14	Timer signal from alarm(1).
SIGTERM	15	Termination signal
SIGSTKFLT	16	Stack fault on coprocessor
SIGCHLD	17	Child terminated
SIGCONT	18	Continue if stopped
SIGTSTOP	19	Stop process
SIGTSTP	20	Stop typed at tty
SIGTTIN	21	tty input for background process
SIGTTOU	22	tty output for background process
SIGIO	23	I/O error
SIGXCPU	24	CPU time limit exceeded
SIGXFSZ	25	File size limit exceeded
SIGVTALRM	26	Virtual time alarm
SIGPROF	27	Profile signal
SIGWINCH	29	Window resize signal

2. Синтаксис та призначення системних викликів

2.1. Системний виклик signal.

```
void (*signal (sig, func)) ( )
                int sig;
                void (*func) ( );
```

Системний виклик signal дозволяє процесу, що викликає вибрати один із трьох можливих способів реакції на одержання визначеного сигналу. Аргументи sig і func специфікують, відповідно, сигнал і вибір.

Аргумент `func` може мати одне з трьох значень: `SI_DFL`, `SI_IGN` чи `адреса_функції`. Макроси `SI_DFL` і `SI_IGN` визначені у файлі `<signal.h>`. Кожний з макросів породжує унікальну константу типу "покажчик на функцію типу `void`".

Дії, що наказуються аргументом `func`, полягають у наступному:

- `SI_DFL` - стандартна реакція на сигнал.

- `SI_IGN` - Ігнорувати сигнал `sig`. Сигнал `SIGKILL` не може ігноруватися.

- `адреса_функції` - перехоплення сигналу. При одержанні сигналу `sig` виконати функцію обробки сигналу `func`; як єдиний аргумент функції `func` передається номер сигналу `sig`; додаткові аргументи передаються для апаратних сигналів. Перед виконанням функції `func` встановлюється стандартна реакція на отриманий сигнал, якщо тільки цей сигнал не є `SIGKILL` чи `SIGTRAP`. Таким чином, щоб перехопити наступний сигнал`sig`, потрібно знову звернутися до `signal`, задавши як аргумент `func` адресу функції обробки.

Після завершення функції обробки сигналу процес, що одержав сигнал, відновляє виконання з точки переривання.

При успішному завершенні системного виклику signal повертається попереднє значення func для зазначеного сигналу sig. У противному випадку повертається значення SI_ERR, а змінній errno присвоюється код помилки. Значення SI_ERR визначене у файлі <sys/signal.h>.

(Примітка: При спробі змінити стандартну реакцію на сигнал `SIGKILL` повертається значення `SI_DFL` (а не `SI_ERR`, як повинно бути), а змінна `errno` одержує значення `EINVAL`.)

2.2. Системний виклик `kill`.

```
int kill (pid, sig)

          int pid

          int sig;
```

Системний виклик `kill` посилає сигнал процесу або групі процесів, що задані ідентифікатором `pid`. Сигнал, що посилається, визначається аргументом `sig` і є одним зі списку сигналів, заданого в системному виклику`signal(2)`, або 0. Якщо аргумент`sig` дорівнює 0 (порожній сигнал), то буде виконуватися перевірка коректності звертання до `kill`, але сигнал у дійсності не посилається. Ця властивість може бути використана для перевірки того, чи правильно заданий аргумент `pid`.

Передаючий і приймаючий процеси повинні мати той самий реальний чи діючий ідентифікатор користувача, якщо тільки діючий ідентифікатор користувача процесу, що посилає, не є ідентифікатором суперкористувача.

При успішному завершенні результат дорівнює 0; у випадку помилки повертається -1, а змінній errno присвоюється код помилки.