Программирование в Linux

Межпроцессное взаимодействие

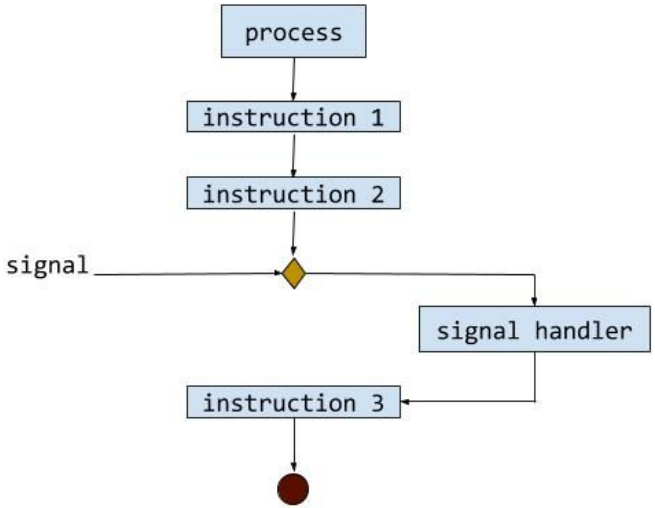
Способы взаимодействия

- файлы
- сигналы
- каналы, FIFO (именованные, неименованные)
- сокеты (сетевые, unix domain socket)
- shared memory (именованная, анонимная)
- очереди событий/сообщений (mqueue)
- ptrace

Способы синхронизации

- файловые мьютексы
- мьютексы (именованные, анонимные)
- семафоры (именованные, анонимные)

Unix-signals




```c
#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);
```

```c
typedef void (*sighandler_t)(int);

sighandler_t signal(int signum, sighandler_t handler);
```

```c
void alarm_handler(int sig) {
    printf("alarm %d\n", sig);
}

int main() {
    signal(SIGALRM, alarm_handler);

    pid_t my_pid = getpid();
    for (int i = 0; i < 5; ++i) {
        printf("kill!\n");
        kill(my_pid, SIGALRM);
    }
    return EXIT_SUCCESS;
}
```

Стандартные сигналы

Signal	Standard	Action	Comment
SIGABRT	P1990	Core	Abort signal from **abort**(3)
SIGALRM	P1990	Term	Timer signal from **alarm**(2)
SIGBUS	P2001	Core	Bus error (bad memory access)
SIGCHLD	P1990	Ign	Child stopped or terminated
SIGCLD	-	Ign	A synonym for **SIGCHLD**
SIGCONT	P1990	Cont	Continue if stopped
SIGEMT	-	Term	Emulator trap
SIGFPE	P1990	Core	Floating-point exception
SIGHUP	P1990	Term	Hangup detected on controlling terminal or death of controlling process
SIGILL	P1990	Core	Illegal Instruction
SIGINFO	-		A synonym for **SIGPWR**
SIGINT	P1990	Term	Interrupt from keyboard
SIGIO	-	Term	I/O now possible (4.2BSD)
SIGIOT	-	Core	IOT trap. A synonym for **SIGABRT**
SIGKILL	P1990	Term	Kill signal
SIGLOST	-	Term	File lock lost (unused)
SIGPIPE	P1990	Term	Broken pipe: write to pipe with no readers; see **pipe**(7)
SIGPOLL	P2001	Term	Pollable event (Sys V). Synonym for **SIGIO**
SIGPROF	P2001	Term	Profiling timer expired
SIGPWR	-	Term	Power failure (System V)
SIGQUIT	P1990	Core	Quit from keyboard
SIGSEGV	P1990	Core	Invalid memory reference
SIGSTKFLT	-	Term	Stack fault on coprocessor (unused)
SIGSTOP	P1990	Stop	Stop process
SIGTSTP	P1990	Stop	Stop typed at terminal
SIGSYS	P2001	Core	Bad system call (SVr4); see also **seccomp**(2)
SIGTERM	P1990	Term	Termination signal
SIGTRAP	P2001	Core	Trace/breakpoint trap
SIGTTIN	P1990	Stop	Terminal input for background process
SIGTTOU	P1990	Stop	Terminal output for background process
SIGUNUSED	-	Core	Synonymous with **SIGSYS**
SIGURG	P2001	Ign	Urgent condition on socket (4.2BSD)
SIGUSR1	P1990	Term	User-defined signal 1
SIGUSR2	P1990	Term	User-defined signal 2
SIGVTALRM	P2001	Term	Virtual alarm clock (4.2BSD)
SIGXCPU	P2001	Core	CPU time limit exceeded (4.2BSD); see **setrlimit**(2)
SIGXFSZ	P2001	Core	File size limit exceeded (4.2BSD); see **setrlimit**(2)
SIGWINCH	-	Ign	Window resize signal (4.3BSD, Sun)

SIGKILL и SIGSTOP нельзя перехватить

```cpp
enum Task {
    kQuit,
    kPrint,
};

struct ThreadSafeQueue {
    std::mutex queue_mutex_;
    std::queue<Task> task_queue_;

    std::optional<Task> Pop() {
        std::scoped_lock lock { queue_mutex_ };
        if (task_queue_.empty()) {
            return std::nullopt;
        } else {
            Task t = task_queue_.front(); task_queue_.pop();
            return t;
        }
    }

    void Push(Task t) {
        std::scoped_lock lock { queue_mutex_ };
        task_queue_.push(t);
    }
} global_queue;

void handle_termination(int sig) {
    if (sig != SIGTERM) {
        return;
    }
    global_queue.Push(kQuit);
}
```

```cpp
int main() {
    signal(SIGTERM, handle_termination);
    global_queue.Push(kPrint);
    std::thread t {
        [](){
            while (true) {
                if (auto it = global_queue.Pop()) {
                    Task t = *it;
                    if (t == kQuit) {
                        printf("quit\n");
                        break;
                    }
                    printf("print!\n");
                } else {
                    global_queue.Push(kPrint);
                    std::this_thread::sleep_for(std::chrono::milliseconds(100));
                }
            }
        }
    };
    t.join();
}
```

ТУТ UB! Обработчик не сигналобезопасен

t: // Pop
scoped_lock {mutex}
!SIGTERM
t: // handle_termination, Push
scoped_lock {mutex}

Блокировка сигналов

```
40  int main() {
41      sigset_t blocked_sigs;
42      sigemptyset(&blocked_sigs);
43      sigaddset(&blocked_sigs, SIGTERM);
44      sigset_t original_mask;
45      pthread_sigmask(SIG_BLOCK, &blocked_sigs, &original_mask);
46      global_queue.Push(kPrint);
47      std::thread t {
48          []{
49              while (true) {...
62          }
63      };
64
65      int recv_sig = 0;
66      if (sigwait(&blocked_sigs, &recv_sig) == 0) {
67          if (recv_sig == SIGTERM) {
68              global_queue.Push(kQuit);
69          }
70      }
71
72      t.join();
73  }
```

```
int ppoll(struct pollfd *fds, nfds_t nfds,
        const struct timespec *tmo_p, const sigset_t *sigmask);

int pselect(int nfds, fd_set *readfds, fd_set *writefds,
            fd_set *exceptfds, const struct timespec *timeout,
            const sigset_t *sigmask);
```

Для однопоточных:

```
/* Prototype for the glibc wrapper function */
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
```

Чуть более продвинутые сигналы: с данными

отправлять:

```
int sigqueue(pid_t pid, int sig, const union sigval value);
```

```
union sigval {
    int   sival_int;
    void *sival_ptr;
};
```

получать (info->si_value):

```
struct sigaction {
    void     (*sa_handler)(int);
    void     (*sa_sigaction)(int, siginfo_t *, void *);
    sigset_t   sa_mask;
    int        sa_flags;
    void     (*sa_restorer)(void);
};
```

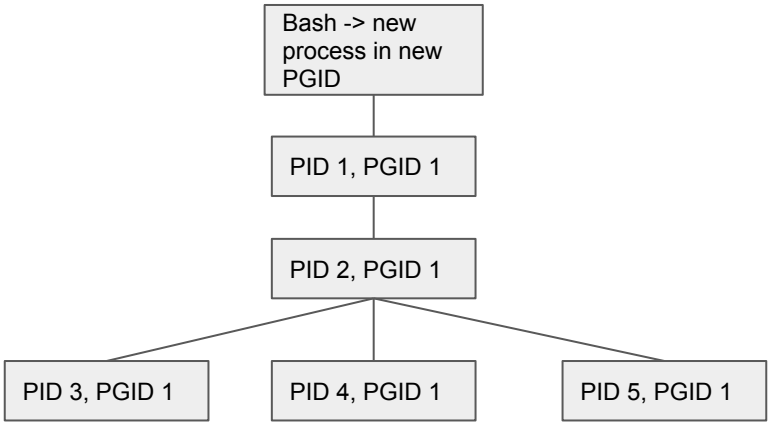
```
int sigaction(int signum, const struct sigaction *act,
              struct sigaction *oldact);
```

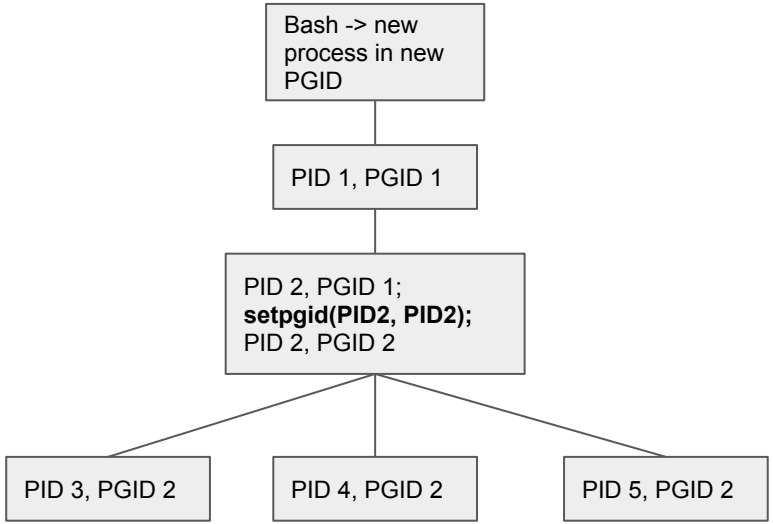
```
#include <signal.h>

int sigwaitinfo(const sigset_t *set, siginfo_t *info);

int sigtimedwait(const sigset_t *set, siginfo_t *info,
                 const struct timespec *timeout);
```

```
9  void alarm_handler(int sig,
10                     siginfo_t* info,
11                     void* ucontext) {
12     printf("alarm %d -- %d\n", sig, info->si_value.sival_int);
13 }
14
15 int main() {
16     struct sigaction act;
17     act.sa_sigaction = alarm_handler;
18     act.sa_flags = SA_SIGINFO;
19     sigemptyset(&act.sa_mask);
20     sigaction(SIGALRM, &act, NULL);
21
22     pid_t my_pid = getpid();
23     for (int i = 0; i < 5; ++i) {
24         printf("kill!\n");
25         sigval_t val = {
26             .sival_int = i
27         };
28         sigqueue(my_pid, SIGALRM, val);
29     }
30     return EXIT_SUCCESS;
31 }
```

broadcast сигналов, process group id (PGID)

kill(-pgid, SIG) — разослать сигнал всем процессам в группе





```c
void alarm_handler(int sig) {
    printf("pid: %d got %d!\n", getpid(), sig);
    SIG_DFL(sig);
}

void child_routine() {
    printf("child=%d, pgrp=%d\n", getpid(), getpgrp());
    pause();
}

int main() {
    signal(SIGALRM, alarm_handler);
    printf("parent=%d, pgrp=%d\n", getpid(), getpgrp());

    pid_t new_group = fork();
    if (new_group > 0) {
        sleep(5);
        // kill group:
        kill(-new_group, SIGALRM);
        return 0;
    }

    printf("group process=%d, pgrp before rebind=%d\n", getpid(), getpgrp());
    // we are in child
    setpgid(getpid(), 0);
    printf("group process=%d, pgrp after rebind=%d\n", getpid(), getpgrp());
    for (int i = 0; i < 5; ++i) {
        pid_t child = fork();
        if (child == 0) {
            child_routine();
            break;
        }
    }
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/signals_egs$ ./sig_broadcast
parent=400284, pgrp=400284
group process=400285, pgrp before rebind=400284
group process=400285, pgrp after rebind=400285
child=400288, pgrp=400285
child=400289, pgrp=400285
child=400290, pgrp=400285
child=400286, pgrp=400285
child=400287, pgrp=400285
pid: 400290 got 14!
pid: 400287 got 14!
pid: 400286 got 14!
pid: 400289 got 14!
pid: 400288 got 14!
```