Семинар по АКОС №8

Interprocess communication

dup

- #include <unistd.h>

- int dup(int oldfd);
- int dup2(int oldfd, int newfd);

- dup – создаёт копию файлового дескриптора oldfd, можно использовать взаимозаменяемо (использует наименьший свободный файловый дескриптор)
- dup2 – использует newfd как копию oldfd: атомарно закрывает newfd + открывает newfd как копию oldfd

- Посмотреть открытые файловые дескрипторы: $ ls -l /proc/*pid*/fd

Pipe

int pipe(int pipefd[2]);

- Неименованный канал
- Один конец на чтение, другой – на запись
- Канал буферизован (64Kb)
- Write:
  - Если при записи не хватает места в буфере, процесс блокируется
  - Если читающий конец закрыт, завершается с ошибкой Broken pipe
- Read:
  - Если буфер пуст, процесс блокируется
  - Если пишущий конец закрыт, возвращает 0

Pipe + fork

- При fork файловые дескрипторы копируются
- Если останутся открытыми дескрипторы на запись, то чтение никогда не завершится

FIFO (именованные каналы)

- int mkfifo(const char *pathname, mode_t mode);

- Как пайп, но с именем
- Как обычный файл, но можно открывать только на чтение или только на запись

Сигналы

Signal	Portable number	Default action	Description
SIGABRT	6	Terminate (core dump)	Process abort signal
SIGALRM	14	Terminate	Alarm clock
SIGBUS	—	Terminate (core dump)	Access to an undefined portion of a memory object
SIGCHLD	—	Ignore	Child process terminated, stopped, or continued
SIGCONT	—	Continue	Continue executing, if stopped
SIGFPE	8	Terminate (core dump)	Erroneous arithmetic operation
SIGHUP	1	Terminate	Hangup
SIGILL	4	Terminate (core dump)	Illegal instruction
SIGINT	2	Terminate	Terminal interrupt signal
SIGKILL	9	Terminate	Kill (cannot be caught or ignored)
SIGPIPE	13	Terminate	Write on a pipe with no one to read it

Отправка сигнала

- #include <sys/types.h>
- #include <signal.h>

- int kill(pid_t pid, int sig);

- Отправить сигнал sig процессу pid

alarm

- #include <unistd.h>
- unsigned int alarm(unsigned int seconds);
- Отправить себе сигнал SIGALRM

Получение сигнала

- При получении сигнала процесс выполняет одно из:

1. Действие по умолчанию
2. Игнорирует сигнал
3. Вызывается обработчик сигнала

Обработчики сигналов

- #include <signal.h>

- int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
- Нельзя поставить обработчик на SIGKILL и SIGSTOP

Маски сигналов

- sigemptyset(sigset_t *set) – создать пустое множество
- sigfillset(sigset_t *set) – заполнить множество
- sigismember(sigset_t *set, int signum) – проверить наличие сигнала
- sigaddset(sigset_t *set, int signum) – добавить сигнал к множеству
- sigdelset(sigset_t *set, int signum) – удалить сигнал из множества

- int sigprocmask(int how, const sigset_t *set, sigset_t *oldset); - поменять маску

Signalfd

- int signalfd(int fd, const sigset_t *mask, int flags);

- Предварительно заблокировать сигналы с помощью sigprocmask