Лекция 16
Межпроцессное взаимодействие

Безопасная обработка сигналов

- Безопаснее всего в обработчике сигнала устанавливать глобальный флаг поступления сигнала, который обрабатывать в основной программе
- Для этого требуются доп. средства управления сигналами

```
volatile sig_atomic_t sigint_flag;
void hnd(int s)
{
  sigint_flag = 1;
}
```

Volatile, sig_atomic_t

- Ключевое слово volatile обозначает, что значение переменной может измениться «неожиданно» для компилятора программы
- Компилятор не должен пытаться оптимизировать обращения к переменной (например, загружая ее на регистр)
- Тип sig_atomic_t — это целый тип (обычно int), для которого гарантируется атомарная запись и чтение

Множества сигналов

// очистка множества

```
void sigemptyset(sigset_t *pset);
```

// заполнение множества

```
void sigfillset(sigset_t *pset);
```

// добавление сигнала в множества

```
void sigaddset(sigset_t *pset, int signo);
```

// удаление сигнала из множества

```
void sigdelset(sigset_t *pset, int signo);
```

Блокирование сигналов

- Если сигнал заблокирован, его доставка процессу откладывается до момента разблокирования

```
int sigprocmask(int how, const sigset_t *set,
                sigset_t *oldset);
```

- SIG_BLOCK — добавить сигналы к множеству блокируемых
- SIG_UNBLOCK — убрать сигналы из множества блокируемых
- SIG_SETMASK — установить множество

Ожидание поступления сигнала

```
int sigsuspend(const sigset_t *mask);
```

- На время ожидания сигнала выставляется множество блокируемых сигналов mask
- После доставки сигнала восстанавливается текущее множество блокируемых сигналов

Стратегия корректной работы с сигналами

- Функции-обработчики сигналов устанавливают флаг поступления сигнала
- Программа выполняется с заблокированными сигналами
- Сигналы разблокируются только на время ожидания прихода сигнала (с помощью sigsuspend или pselect)

Именованные каналы

- Канал, доступ к которому выполняется через точку привязки файловой системы
- Ядро создает по одному объекту именованного канала для каждой записи в файловой системе

```
int mkfifo(const char *path, mode_t mode);
```

- Создание специального объекта в файловой системе
- Удаление — с помощью unlink

Открытие именованного канала

- Открывается с помощью системного вызова open:

```
fdr = open(path, O_RDONLY, 0); // на чтение
fdw = open(path, O_WRONLY, 0); // на запись
```

- Операции открытия блокируются до выполнения противоположной операции
  - Открытие на чтение заблокирует процесс, пока другой процесс не откроет на запись
  - Открытие на запись заблокирует процесс, пока другой процесс не откроет на чтение
- После открытия работа — как с обычным каналом

Открытие канала

- Допускается открытие именованного канала в режиме O_RDWR, тогда канал откроется независимо от наличия читателей.

```
fdw = open(path, O_RDWR, 0);
fdr = open(path, O_RDONLY, 0);
```

- Таким образом можно полностью открыть канал в одном процессе

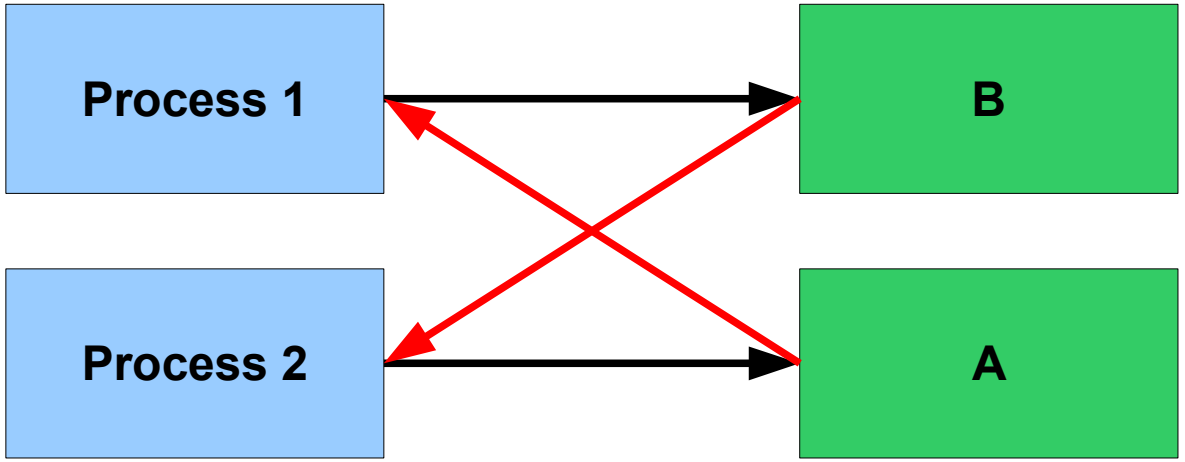
Типичные ошибки во взаимодействующих процессах

- Deadlock (тупик) — несколько процессов не может продолжить выполнение, так как процессы ждут друг друга
- Race Condition (гонки) — результат работы зависит от порядка переключения выполнения между параллельными процессами
- **Очень сложно обнаруживаемые ошибки**
- **Могут проявляться очень редко при редкой комбинации условий**

Обнаружение тупиков

Process 1:
wait(&A);
notify(&B);

Process 2:
wait(&B);
notify(&A);



- Захваченный ресурс — дуга от процесса к ресурсу
- Ожидаемый ресурс — дуга от ресурса к процессу
- Если в графе есть цикл, система попала в состояние тупика