Семинар 21. Сигналы

Сигналы

- **Сигнал** - программное прерывание нормальной работы процесса
- Некоторые сигналы сообщают об ошибках, таких как недопустимое обращение к памяти, другие сообщают об асинхронных событиях, таких как потеря связи с терминалом
- Некоторые события делают нежелательным или невозможным продолжение нормальной работы процесса, такие сигналы по умолчанию завершают работу процесса. Другие сигналы, сообщающие о «безобидных» событиях, по умолчанию игнорируются
- Процесс может послать сигнал другому процессу. Это позволяет, например, родительскому процессу завершить выполнение дочернего процесса

Типы сигналов

- **Ошибка** означает, что программа сделала что-то недопустимое и не может продолжить своё выполнение (например, деление на 0)
- **Внешнее событие** обычно относится к операциям ввода/вывода или другим процессам (например, получение данных при асинхронных операциях)
- **Явный запрос** предполагает использование функций, таких как kill, задача которых — сгенерировать сигнал

Типы сигналов

- **Синхронные** - относится к некоторому действию в программе и генерируется и доставляется (если не заблокирован) во время этого действия (например, большинство ошибок, сигнал, посланный самому себе)
- **Асинхронные** - могут приходить в непредсказуемые моменты времени

Генерация сигнала

- Системный вызов **kill**
- Если установить номер сигнала равным **0**, то сигнал не посылается, но все ошибки проверяются. Это можно использовать, например, для проверки существования процесса с заданным **pid**
- **int raise(int sig);** эквивалентно **kill(getpid(), sig);**

Получение сигнала

- После генерации сигнала он становится ожидающим доставки, обычно - непродолжительное время
- Сигнал может быть заблокированным, тогда он будет ожидать разблокировки, после чего будет доставлен немедленно
- Получение сигнала означает выполнение определенного действия. Для некоторых (**SIGKILL** и **SIGSTOP**) действие фиксировано, но для большинства есть выбор: игнорировать, обрабатывать или действовать по умолчанию

Получение сигнала

- Программа может задать обработчик сигнала с помощью **signal** или **sigaction**
- Если для некоторого типа сигнала установлено игнорирование, любой сигнал такого типа будет сброшен сразу же после генерации, даже если этот тип сигналов в тот момент был заблокирован. Такой сигнал никогда не будет доставлен, даже если программа затем установит обработчик сигнала и разблокирует его

Получение сигнала

- Если не установлен обработчик, и сигнал не игнорируется, то выполняется действие по умолчанию
- Для большинства сигналов - завершить исполнение программы, для некоторых безобидных - не делать ничего
- Сигналы, по умолчанию завершающие процесс, вызывают функцию ядра, аналогичную **_exit**, то есть не вызываются обработчики завершения, зарегистрированные по **atexit**, не записываются буферы дескрипторов потока **FILE**

Стандартные сигналы

- Имена сигналов определены в заголовочном файле **signal.h**
- Макрос **NSIG** даёт общее количество сигналов, определённых в системе. Поскольку сигналы нумеруются последовательно, его значение на 1 больше максимального номера сигнала
- Для получение строки, описывающей сигнал, можно использовать **strsignal**

Стандартные сигналы

- Когда процесс остановлен, он не может получать сигналы, кроме **SIGKILL** и **SIGCONT**. Все посланные процессу сигналы будут сделаны ожидающими доставки
- Когда процесс получает сигнал **SIGCONT**, все сигналы остановки, ожидающие доставки, будут сброшены. Аналогично, когда процесс получает сигнал остановки, все сигналы **SIGCONT**, ожидающие доставки, будут сброшены

Работа с множествами сигналов

- Аргументами многих функций, работающих с сигналами, могут быть множества сигналов
- Тип **sigset_t** должен использоваться для хранения множества сигналов
- Функции для работы с множествами сигналов: **sigemptyset**, **sigfillset**, **sigaddset**, **sigdelset**, **sigismember**

Установка обработчика сигнала

- Если процесс игнорирует сигнал **SIGSEGV** и другие аналогичные сигналы, его поведение после ошибки неопределено
- Игнорировать запросы пользователя, такие как **SIGINT** и пр. — недружественно по отношению к пользователю
- Функция **signal** присутствует в стандарте ANSI C, тем не менее её использование не рекомендуется по историческим причинам
- sigaction - предпочтительный вариант

Блокирование сигналов

- Временная блокировка сигнала с помощью **sigprocmask** позволяет предотвратить прерывание нормальной работы процесса в критической секции кода
- Сигнал будет доставлен после его разблокировки

Ожидание прихода сигналов

- **pause** приостанавливает выполнение процесса до поступления сигнала, который не блокируется и не игнорируется
- Если поступление сигнала запускает функцию обработки сигнала, которая возвращает управление в процесс, функция pause возвращается с кодом завершения **-1** и кодом ошибки **EINTR** даже в случае, когда включена семантика перезапускаемых системных вызовов
- Однако, с **pause** можно получить ошибки обработки сигналов, связанные с асинхронностью их природы

Ожидание прихода сигналов

- Более правильный способ - использование **sigprocmask** совместно с **sigsuspend**
- временно устанавливает маску блокировки сигналов на переданную ему, а затем ждет сигнал, который либо вызывает завершение процесса, либо обрабатывается процессом