Inter Process Communication

Возникновение двух коммерчески значимых линий: System5 и UNIX BSD. Они были приватизированы фирмами. В результате возникла ситуация: ПО для System5 нельзя было выполнять в UNIX BSD и наоборот. То есть, поскольку данные системы стали лицензионными, разрабам пришлось покупать лицензии. Возникла некоторая монополия. Пишут, что сообщество программистов обратилось с идеей создания станадарта, но все равно: POSIX - государственный стандарт.

POSIX (Portable Operating System Interface for UNIX) перечисля более 1000 системных вызовов, обеспечивая тем самым переносимость ПО.

!!! POSIX оговаривает только системные вызовы. Структура ядра и его функции не оговорены.

Факт: IBM, когда разрабатывалась серия машин IBM360, разделило стоимость «харда» и «софта». В это же время в СССР была другая тенденция: поднять цену. А у «капиталюг» - наоборот. Почему? На самом деле на рынке важно количество продаж.

FIPS (Federal Information Processing Standard) - [кем разработан]

POSIX.1 был разработан в 1988 году. Потом POSIX.2, но этого было все равно мало.

В Европе был разработан стандарт X/Open.

Это важно для нас, так как рынок ПО получил определенную свободу. Все это позволяет более эффективно разрабатывать ПО, привелкать больше людей к разработке.

Inter Process Communication System5 включает в себя: сигналы, семафоры, программные каналы (им. и неим.), очереди сообщений, сегменты разделяемой памяти. Есть еще RPC [...].

Сигналы

Сигналы информируют процессы и группы процессов.

Сигнал - это программное средство информированния процессов о событиях, которые могут происходить внутри процессов (**синхронные** события) и вне процессов (**ассинхронные** события).

Все процессы, которые выполняются в системе, являются ассинхронными (выполняются с собственной скоростью независимо от того, как выполняются другие процессы).

Как правило, получение некоторым процессом сигнала указывает ему завершить свое выполнение. Реакция процесса на получаемый сигнал зависит от того, как процесс определяет свою реакцию на получаемый сигнал: проигнорировать, реагировать на получаемый сигнал по умолчанию (так, как определено в системе) или определить собственную реакцию на получаемый сигнал (для этого в коде процесса должен быть написан собственный обработчик сигнала).

В классическом UNIX не могло быть больше 20 сигналов.

```
#define NSIG 20
#define SIGHUP 1
#define SIGINT 2
 ...
#define SIGKILL 9
 ...
#define SIGSEGV 11
#define SIGSYS 12
#define SIGPIPE 13
#define SIGALARM 14
#define SIGTERM 15
```

```
12 #define SIGUSR1 16
13 #define SIGUSR2 17
14 #define SIGCLD 18
15 #define SIGPWR 19
```

- SIGHUP - сигнал разрыва с терминалом
- SIGINT - CTRL+C (завершение процесса)
- SIGKILL - уничтожение процесса системным вызовом kill
- SIGSEGV - нарушение сегментации (выход за пределы сегмента)
- SIGSYS - ошибка выполнения системного вызова
- SIGPIPE - запись в неименованный программный канал есть, чтения - нет
- SIGALARM - сигнал побудки
- SIGTERM
- SIGUSR1, SIGUSR2
- SIGCLD (в Linux - SIGCHLD) - сопровождает завершение процесса потомка
- SIGPWR - отключение питания (Для сохранения своей работоспособности ОС выполняет ряд действий: сохраняет важнейшие данные. При этом сигнал может перехватываться приложением)

О нарушениии сегментации: [картинка из методички про fork]. А в классическом UNIX было сегментированное управление памятью (???).

Средством посылки сигналов в ОС UNIX являются два системных вызова: kill и signal.

Системный вызов kill

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

```
1 kill(getpid(), SIGALARM);
```

Сигнал побудки будет послан процессу, вызвавшему kill.

!!! Для винды основным документом является MSDN. Для Linux - мануал.

Если параметр $pid \leq 1$, то сигнал будет послан группе процессов: если 0, то всем процессам с идентификатором группы, совпадающим с идентификатором группы процесса, который вызвал kill; если -1, то sig будет послан процессам, которые имеют тот же uid, что и процесс, вызвавший kill и т.д.

Системный вызов signal

```
1 void (*signal(int sig, void (*handler)(int)))(int);
```

signal не входит в POSIX, но входит в ANSI C. Поэтому signal не рекомендуется использовать в переносимых системах (аналог - sigaction).

```c
#include <signal>

int main()
{
  void (*old_handler)(int) = signal(SIGINT, SIG_IGN);
  ...
  signal(SIGINT, old_handler);
  ...
}
```

sigaction

```c
int sigaction(int sig_numb, struct sigaction *action, struct sigaction *
   old_action);
```

В POSIX имеется два системных вызова: sigsetjump и siglongjmp [что делают].

Программные каналы

mknod создает именованный программный канал. pipe - неименованный (не имеют имени, но имеют дескриптор struct inode).