Продолжение проблемы ввода - вывода.

Ввод-вывод, управляемый сигналами и асинхронный ввод - вывод.

Эти модели ввода-вывода относятся к асинхронным; мультиплексирование - асинхронный блокирующий ввод - вывод;

асинхронный ввод - вывод предполагает либо сокращение времени блокировки, либо полное отсутствие блокировки процесса в ожидании завершения ввода - вывода.

Это связано с тем, что блокировки замедляют выт-е процесса (негативный фактор);

От блокировок стремятся уйти там, где это возможно;

В совр. системах для отдельно стоящей машины блокирующий ввод - вывод - основная модель ввода - вывода (запросив ввод-вывод, приложение блокируется, система переходит в режим ядра и запрос обслуживается);

При этом в блокирующем вводе-выводе в полной мере реализована идея распараллеливания ф-ций (3-е поколение ЭВМ), которая базируется на стремлении освободить процессор от непроизводительных действий (упр-я медленными внеш. устр-вами);

Пока медленное внеш. устр-во выполняет задачу ввода данных, процессор может перейти на вып-е другой работы;

В состав ЭВМ были включены каналы (программно управляемых устр-ва);

Эта же идея нашла отражение в шинной архитектуре ПК;

В канальной архитектуре внеш. устр-вами управляют каналы, а в шинной – контроллеры (также программно управляемые устр-ва);

Появление блокирующего ввода-вывода связано с задачей распараллеливания ф-ций, решённой в 3 поколении ЭВМ передачей ф-ций упр-я внеш. устр-вами спец. процессорам – программно управляемым устр-вам;

Также необходимо информировать процессор о завершении операции ввода-вывода;
Это привело к появлению аппаратных прер-й;
В 3 поколении ЭВМ возникла полноценная система прер-й, существующая и в совр. компьютерах;

Аппаратные прер-я неоднородны: отдельно рассм. прер-е от сист. таймера (единственное периодическое прер-е в системе), а все остальные прер-я по сути явл. прерываниями от внеш. устр-в;

Задача прерывания от внеш. устр-в –

- проинформировать процессор о завершении операции ввода-вывода.

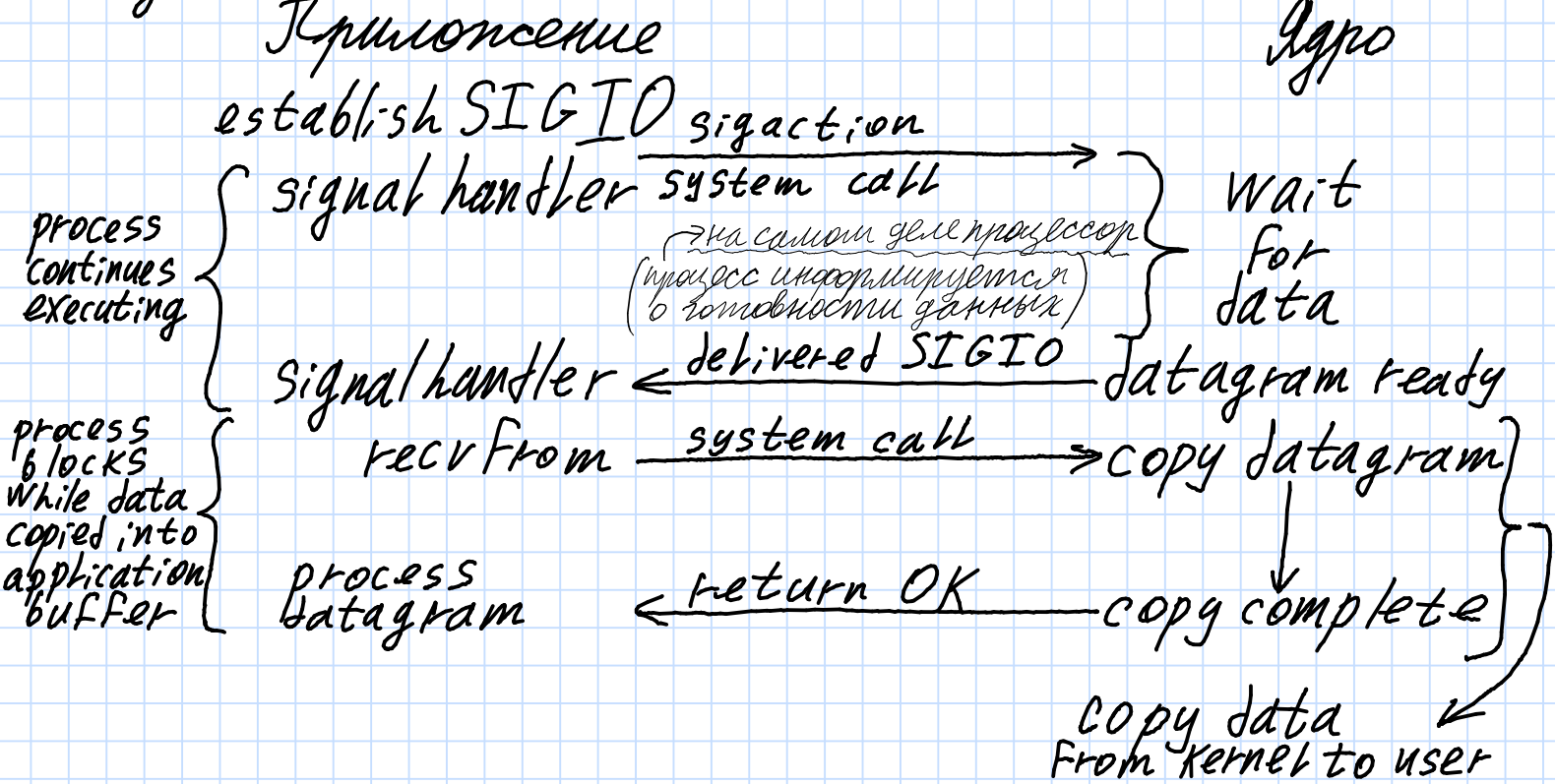
В простейшем варианте это выполняется через контроллер прерываний: (PIC, см. схему из л/р №2)
Когда заканчивается операция ввода-вывода, контроллер устр-ва посылает на контроллер прер-й сигнал, получив который контроллер прер-й формирует сигнал INT, который посылает на ногу процессора.
Процессор в конце цикла вып-я каждой команды проверяет наличие этого сигнала. Если сигнал пришёл, он посылает ответный сигнал. Получив его, контроллер прер-й выставляет на шину данных вектор прерывания (в DOS - смещение к вектору прер-я в таблице векторов прер-й)

вектор прерывания - адрес обработчика прерывания;

Это делается для того, чтобы процессор перешёл на выполнение кода обработчика прерывания;

④ Ввод-вывод, управляемый сигналом.

Модель:



Устанавливается обработчик спец. сигнала SIGIO, задача которого - проинформировать процесс о том, что данные готовы (delivered SIGIO - доставка сигнала процессу);

Процесс вызывает receive from, чтобы получить данные;

Буфер устр-ва → Буфер ядра → Буфер приложения;

Процесс продолжает выполняться и блокируется только на время копирования данных, т.е. время блокировки уменьшилось;

(ужасная формулировка)

„Маленькая крит. секция процесса" = код выполняется быстро, т.е. блокировка занимает мало времени;

Мы рассм. модели ввода-вывода с точки зрения программиста, не вникая в тонкости аппаратной реализации;

Всю работу в данном способе ввода-вывода берёт на себя ядро: оно отслеживает готовность данных и после этого посылает сигнал SIGIO. В рез-те будет вызван обработчик этого сигнала;

Это называется callback-функцией (функцией обратного вызова)

При этом receive from можно вызвать либо в обработчике сигнала, либо в главном потоке программы;

Сигнал типа SIGIO для каждого процесса м.б. только один;

В рез-те в каждый момент времени можно работать только с одним файловым дескриптором;

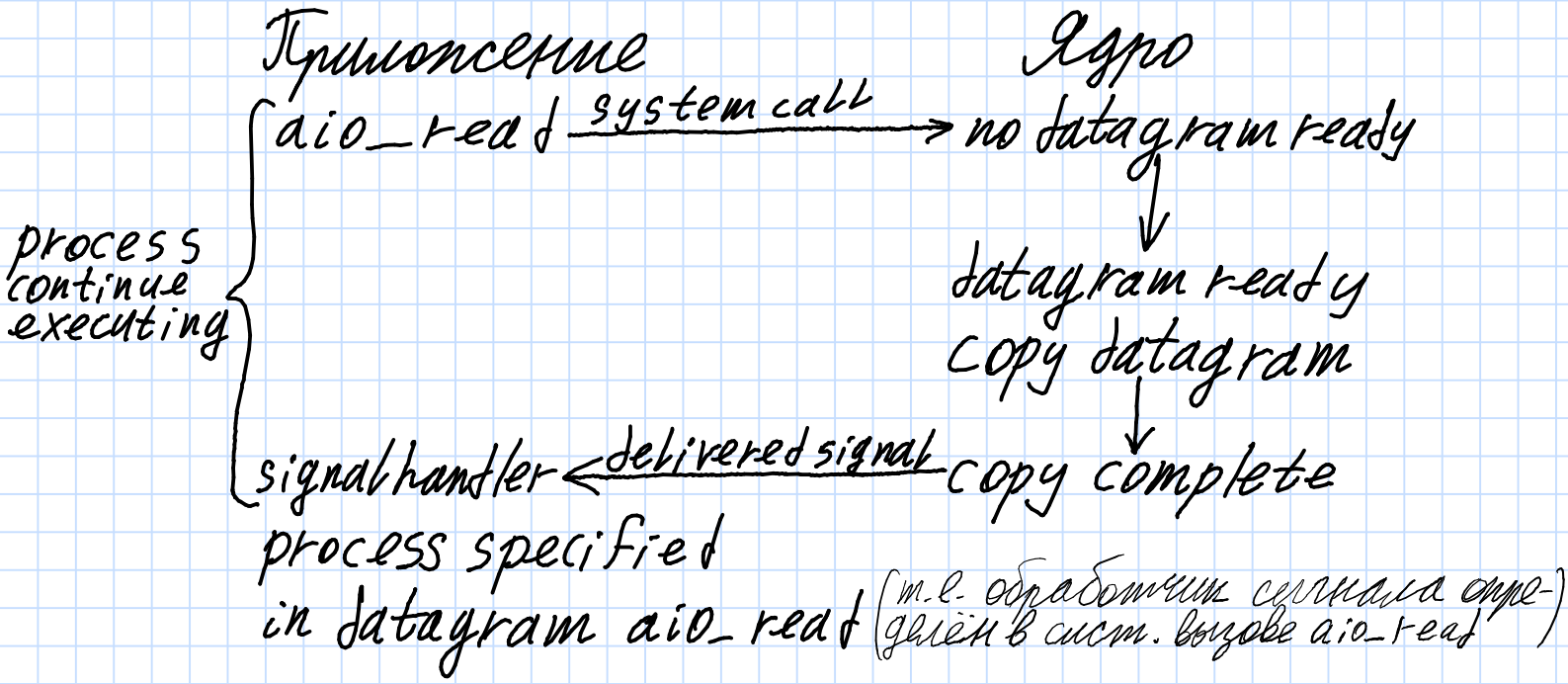
На время выполнения обработчика сигнала данный сигнал блокируется.

Если в период блокировки доставляется несколько сигналов, то они <u>теряются</u>;

Если маска сигнала (sa_mask) установлена в NULL, то на время выполнения обработчика другие сигналы не блокируются;

(5) Асинхронный ввод - вывод.

Для его реализации система предоставляет пользователю спец. сист. вызовы (как правило, начинаются с „a" - asynchronous)



Процесс не блокируется и продолжает выполняться. В отличие от пред. модели, сигнал информирует процесс о <u>полном завершении операции ввода-вывода</u>, включая копирование данных;

Здесь вместо receive from исп. read/write, т.к. здесь нет передачи сообщений, это в чистом виде ввод-вывод на отдельно стоящей машине;

receive from/send to - сист. вызовы, связанные с приёмом/посылкой сообщений;
read/write - сист. вызовы, связанные с получением данных от внеш. устр-в (по сути из файла, т.к. в UNIX всё файл);

Асинхронный ввод-вывод определён в POSIX (спецификация POSIX, согласовавшая все различия в ф-циях асинхронного ввода-вывода, возникших в разных системах, объединив их достоинства (описаны в мануале))
Все ф-ции асинхронного ввода/вывода работают таким образом, что они сообщают ядру о начале операции ввода-вывода, а процесс они уведомляют о завершении операции ввода-вывода, включая копирование данных;

Основное отличие асинхронного ввода-вывода от ввода-вывода, управляемого сигналом:
При вводе-выводе, управляемом сигналом, сигнал информирует процесс о готовности данных, а при асинхронном вводе-выводе процесс вообще не блокируется и информируется о полном завершении операции ввода-вывода;

Проблема асинхронного ввода-вывода: необходимо получать асинхронные события синхронно, т.к. данные нужны приложению для выполнения дальнейших действий;

В асинхронном вводе-выводе внимание сосредотачивается на 2 моментах:

1) на возможности определить, что ввод-вывод можно выполнить быстро (в технике нет быстро/медленно);
2) на завершении операции ввода-вывода в случае невозможности немедленного выполнения ввода-вывода, т.е. возвращения ошибки;

Системы реального времени работают с готовыми данными (с датчиков и измерит. приборах, считавших показания). Данные готовы, их нужно только получить. Если данные не готовы, будет возвращена ошибка, т.е. мы не можем обработать соотв. данные, и надо выполнить доп. действия. Надо понимать когда это возможно, а когда нет;

Классификация:

	Blocking	Non-blocking
Synchronous	① recvfrom/sendto (↓ разных источниках по-разному) read/write	② polling (опрос)
Asyncronous	③ IO multiplexing ④ SIGIO	⑤ AIO