Домашняя работа №3 (2015)

Problem H31: Конвейер-1

Напишите программу, которая получает следующие параметры командной строки: `file prog1 prog2 prog3 ...` (многоточие означает, что возможны дополнительные параметры; словами перечислены обязательные параметры) и моделирует следующую команду shell:
`prog1 < file ; prog2 | prog3 ...`

Операция | означает конвейер. `p1 | p2` означает, что `p1` и `p2` запускаются одновременно и стандартный вывод `p1` подается на стандартный ввод `p2`. Операция ; означает последовательное выполнение. `p1 ; p2` означает, что сначала выполняется `p1` и только после его завершения (или если `p1` не запустилась из-за ошибки) выполняется `p2`. Приоритет операции | выше приоритета операции ;

Программа должна завершаться с кодом 2, если у нее меньше 4 параметров. При этом на стандартном потоке ошибок должно печататься сообщение `Bad arguments`.

Программа должна завершаться с кодом 1, если при выполнении этой команды произошла ошибка, фатально прервавшая выполнение этой команды, на стандартном потоке ошибок должна быть написана причина ошибки при помощи стандартной функции `strerror`.

Иначе программа должна завершаться с кодом 0.

В решении запрещается передавать управление программам-shell'ам (в частности, при помощи функции `system`). Необходимо "вежливо" завершать программы и явно освобождать занимаемые ресурсы (файлы, динамическую память и т.п.). В таблице процессов не должны оставаться "зомби" по окончании работы программы. Ожидание процессов должно быть пассивным, т.е. не тратить ресурсы процессора. Последним должен завершаться отцовский процесс.

Можно предполагать, что программы `prog1`, `prog2` и `prog3` не возвращают код 127 при завершении. Можно предполагать, что отцовский процесс запускается в своей группе процессов.

Problem H32: Синхронное чтение

Напишите программу, которая обрабатывает текстовый ввод (со стандартного потока ввода) при помощи параллельных процессов указанным ниже образом.

Программа создает два сыновних процесса (назовем их "первым" и "вторым"). Первый процесс обрабатывает нечетные строки ввода, второй - четные строки ввода. Строки нумеруются с единицы. Первый процесс делает регистр каждой буквы строки верхним (a -> A, A -> A, 2 -> 2, # -> #), второй процесс - нижним (a -> a, A -> a, 2 -> 2, # -> #). Изменению регистра подвергаются только латинские символы. Каждый процесс сразу печатает на стандартный поток вывода полученную строку. По окончании ввода первый и второй процесс завершаются, а с ними завершается и отцовский процесс.

Необходимо реализовать следующее взаимодействие процессов. В то время, когда один сыновний процесс обрабатывает очередную строку, другой сыновний процесс должен ожидать окончания этой обработки. После успешного чтения и обработки строки сыновний процесс передает управление своему брату (например, так: сыновний процесс уведомляет отцовский процесс сигналом `SIGUSR1`, а отцовский процесс пересылает этот сигнал другому сыновнему процессу и тот начинает работу над следующей строкой). Если сыновний процесс не смог считать строку (ошибка при чтении или ввод был весь прочитан ранее), он посылает отцовскому процессу сигнал `SIGUSR2` и тот завершает все сыновние процессы, посылая им сигнал `SIGTERM` и ожидая их завершения. Отцовский процесс завершается с кодом 0, если не было ошибок при чтении.

Если при чтении ввода произошла ошибка, отцовский процесс получает от сыновнего процесса сообщение об ошибке, завершает сыновние процессы, отправляя им `SIGTERM` и ожидая их завершения, пишет на стандартный поток ошибок полученное сообщение об ошибке и завершается с кодом 1. Сообщение должно быть получено при помощи стандартной функции `strerror`.

Необходимо явно освобождать занимаемые ресурсы (файлы, динамическую память и т.п.). В таблице процессов не должны оставаться "зомби". Ожидание процессов должно быть пассивным, т.е. не тратить ресурсы процессора.

Примечание: задача решается проще, если процессы будут разделять одну запись системной файловой таблицы.

Можно предполагать, что отцовский процесс запускается в отдельной группе процессов.

Problem H33: Посекундно

Напишите программу, получающую первым параметром командной строки имя файла, а вторым аргументом - целое положительное число `N` (других параметров быть не должно). Программа должна открыть файл для чтения, читать из него целые числа в бинарном виде до конца и вычислять максимальное число в этом файле. После чтения очередного числа программа сразу должна переходить к чтению следующего числа. Программа может предполагать, что ей подается файл только из целого числа байтов.

Каждые `N` секунд с момента своего старта программа должна печатать строку на стандартный вывод, являющуюся текущим найденным максимумом (строка должна быть текстовым представлением текущего максимума). При нажатии `Ctrl+C` программа должна завершаться с кодом 1 и печатать найденный максимум на момент завершения. Если же программа завершилась не по нажатии `Ctrl+C`, а по окончании файла, то она тоже печатает найденное максимальное значение и завершается с кодом 0.

Если программа не получила нужное количество параметров или второй параметр не является целым положительным числом, программа завершается с кодом 2 и пишет строку `Bad arguments` на стандартный поток ошибок.

Если программа получила параметры и второй параметр является числом, но не смогла открыть файл, она завершается с кодом 3 и пишет строку, являющуюся причиной проблемы с открытием файла, полученную при помощи функции `strerror`, на стандартный поток ошибок.

Если поданный файл пуст, программа должна завершается с кодом 4 и пишет строку `Empty file` на стандартный поток ошибок.

Необходимо "вежливо" завершать программу и явно освобождать занимаемые ресурсы (файлы, динамическую память и т.п).

Можно предполагать, что отцовский процесс запускается в своей группе процессов.

Problem H34: Переключатель

Напишите программу, которая получает в командной строке два параметра-строки (других параметров у программы не должно быть). Каждая из этих строк является строкой запуска некоторой программы (например, "`gcc myprog.c -o myprog`").

Программа должна запустить эти программы в сыновних процессах так, чтобы в каждый момент времени работал только один из них (другой процесс в это время должен быть приостановлен). Переключение на другой сыновний процесс производится только по отправке сигнала `SIGQUIT` отцовскому процессу (например, такой сигнал генерируется при нажатии сочетания клавиш `Ctrl+\`). Если же этот сигнал приходит более 1 раза в течение двух секунд ("быстрое нажатие"), выполнение обоих сыновних процессов завершается при помощи посылки им соответствующего сигнала, а отцовский процесс завершается с кодом 0 после завершения сыновних.

Если в какой-то момент оба сыновних процесса завершились до "быстрого нажатия", отцовский процесс тоже должен завершиться с кодом 0 сразу после завершения сыновей.

Если в какой-то момент сам завершился один из сыновних процессов (до "быстрого нажатия"), а другой сыновний процесс еще работает, отцовский процесс не должен реагировать на одиночное поступление `SIGQUIT`, а при "быстром нажатии" сыновний процесс должен быть завершен посылкой соответствующего сигнала (если, конечно, он не завершился к этому моменту сам), после чего отцовский процесс тоже завершается с кодом 0.

Если программа не получила нужное количество параметров, она завершается с кодом 2 и пишет строку `Bad arguments` на стандартный поток ошибок.

Если программа получила параметры, но не смогла запустить хотя бы одну из программ, она завершается с кодом 1 (ошибка обнаруживается в момент выполнения сыновнего процесса). Если в момент обнаружения ошибки запуска программы, уже создан другой сыновний процесс, он должен быть завершен посылкой соответствующего сигнала. Если сыновний процесс не создан, то он не создается. Отцовский процесс должен написать строку, являющуюся причиной проблемы с запуском программы, полученную при помощи функции `strerror`, на стандартный поток ошибок.

В решении запрещается передавать управление программам-shell'ам (в частности, при помощи функции `system`). Необходимо "вежливо" завершать программы и явно освобождать занимаемые ресурсы (файлы, динамическая память и т.п.). В таблице процессов не должны оставаться "зомби". Ожидание процессов должно быть пассивным, т.е. не тратить ресурсы процессора.

Можно предполагать, что отцовский процесс запускается в своей группе процессов.

Problem H35: Конвейер-2

Напишите программу, которая последовательно считывает со стандартного ввода строки (обозначим их `prog1`, `prog2`, ..., `progN`) до конца ввода. Каждая строка - это имя некоторой программы, директория которой находится в переменной окружения `PATH`, или абсолютный путь к некоторой программе (например, "/usr/bin/gcc"). Длина строки заранее неизвестна. После окончания чтения программа должна выполнить следующую команду shell:

```
prog1 | prog2 | ... | progN
```

В начале и конце строк возможны незначащие символы (пробелы и символы табуляции), они не входят в путь к программе.

В случае отсутствия в очередной строке пути к программе ваша программа должна завершаться с кодом 1 и писать строку `Bad input` на стандартный поток ошибок. При этом все запустившиеся процессы должны быть завершены посылкой соответствующего сигнала, а чтение должно быть прервано.

В случае наличия в очередной строке пути к программе, но ошибке при запуске этой программы, ваша программа должна завершаться с кодом 2 и писать причину ошибки, полученную при помощи стандартной функции `strerror`, на стандартный поток ошибок. При этом все запущенные сыновние процессы должны быть завершены посылкой им соответствующего сигнала, а чтение должно быть прервано.

В случае возникновения других ошибок ваша программа должна завершаться с кодом 3 и писать причину ошибки, полученную при помощи стандартной функции `strerror`, на стандартный поток ошибок. При этом все запущенные сыновние процессы должны быть завершены посылкой им соответствующего сигнала, а чтение должно быть прервано.

Иначе ваша программа должна завершаться с кодом возврата программы `progN` или 0, если ввод был пустым.

В решении запрещается передавать управление программам-shell'ам (в частности, при помощи функции `system`). Необходимо "вежливо" завершать программы и явно освобождать занимаемые ресурсы (файлы, динамическая память и т.п.). В таблице процессов не должны оставаться "зомби". Ожидание процессов должно быть пассивным, т.е. не тратить ресурсы процессора.

Ваше решение должно использовать минимально возможное число каналов. Можно считать, что весь ввод укладывается в оперативную память. Все процессы в конвейере должны запускаться параллельно. Последним должен завершаться отцовский процесс.

Можно предполагать, что отцовский процесс запускается в своей группе процессов.