СОЗДАНИЕ ЗАВЕРШЕНИЕ И СОСТОЯНИЯ ПРОЦЕССОВ

1. СОЗДАНИЕ

Операционным системам необходим какой-нибудь способ для создания процессов. В самых простых системах или в системах, сконструированных для запуска только одного приложения (например, в контроллере микроволновой печи), появляется воз- можность присутствия абсолютно всех необходимых процессов при вводе системы в действие. Но в универсальных системах нужны определенные способы создания и прекращения процессов по мере необходимости.

Существуют четыре основных события, приводящих к созданию процессов.

1. Инициализация системы. (запуск)

При запуске операционной системы создаются, как правило, несколько процессов. Некоторые из них представляют собой высокоприоритетные процессы, то есть про- цессы, взаимодействующие с пользователями и выполняющие для них определенную работу. Остальные являются фоновыми процессами, не связанными с конкретными пользователями, но выполняющими ряд специфических функций. Например, фо- новый процесс, который может быть создан для приема входящих сообщений элек- тронной почты, основную часть времени проводит в спящем режиме, активизируясь только по мере появления писем. Другой фоновый процесс, который может быть создан для приема входящих запросов на веб-страницы, размещенные на машине, просыпается при поступлении запроса с целью его обслуживания. Фоновые процес- сы, предназначенные для обработки какой-либо активной деятельности, связанной, например, с электронной почтой, веб-страницами, новостями, выводом информации на печать и т. д., называются **демонами**. Обычно у больших систем насчитываются

десятки демонов. В UNIX1 для отображения списка запущенных процессов может быть использована программа ps. В Windows для этой цели может использоваться диспетчер задач.

1. Выполнение работающим процессом системного вызова, предназначенного для создания процесса.

Вдобавок к процессам, созданным во время загрузки, новые процессы могут быть соз- даны и после нее. Часто бывает так, что работающий процесс осуществляет системный вызов для создания одного или более новых вспомогательных процессов. Создание новых процессов особенно полезно, когда выполняемая работа может быть легко вы- ражена в понятиях нескольких связанных друг с другом, но в остальном независимых друг от друга взаимодействующих процессов. Например, если из сети выбирается большой объем данных для последующей обработки, наверное, будет удобно создать один процесс для выборки данных и помещения их в общий буфер, чтобы в то же са- мое время второй процесс забирал элементы данных и проводил их обработку. Также можно ускорить выполнение работы, если на многопроцессорной системе разрешить каждому процессу работать на разных центральных процессорах.

1. Запрос пользователя на создание нового процесса.

В интерактивных системах пользователи могут запустить программу вводом команды или щелчком (двойным щелчком) на значке. Любое из этих действий дает начало ново- му процессу и запускает в нем выбранную программу. В основанных на применении команд UNIX-системах с работающей X-оболочкой новый процесс получает окно, в котором он был запущен. При запуске в Microsoft Windows процесс не имеет окна, но он может создать одно или несколько окон, и в большинстве случаев так и проис- ходит. В обеих системах пользователи могут одновременно открыть несколько окон, в каждом из которых запущен какой-нибудь процесс. Используя мышь, пользователь может выбрать окно и взаимодействовать с процессом, например, если потребуется, вводить данные.

1. Инициация пакетного задания.

Последнее событие, приводящее к созданию процесса, применимо только к системам пакетной обработки данных, имеющимся на больших универсальных машинах. Пред- ставьте себе управление запасами товаров в конце рабочего дня в сети магазинов. Здесь пользователи могут отправлять системе пакетные задания (возможно, с помощью уда- ленного доступа). Когда операционная система решает, что у нее достаточно ресурсов для запуска еще одного задания, она создает новый процесс и запускает новое задание из имеющейся у нее очереди входящих заданий.

1. ЗАВЕРШЕНИЕ

После создания процесс начинает работать и выполняет свою задачу. Но ничто не длит- ся вечно, даже процессы. Рано или поздно новые процессы будут завершены, обычно в силу следующих обстоятельств:

-обычного выхода (добровольно);

Большинство процессов завершаются по окончании своей работы. Когда компилятор откомпилирует заданную ему программу, он осуществляет системный вызов, сообща- ющий операционной системе о завершении своей работы. Этим вызовом в UNIX яв- ляется *exit*, а в Windows — *ExitProcess*. Программы, работающие с экраном, также поддерживают добровольное завершение. Текстовые процессоры, интернет-браузеры и аналогичные программы всегда содержат значок или пункт меню, на котором поль- зователь может щелкнуть, чтобы приказать процессу удалить все временные файлы, которые им были открыты, и завершить свою работу.

- выхода при возникновении ошибки (добровольно);

Вторая причина завершения — обнаружение процессом фатальной ошибки. Например, если пользователь наберет команду

cc foo.c

с целью компиляции программы foo.c, а файла с таким именем не будет, то произойдет простое объявление о наличии данного факта и выход из компилятора. Выхода из ин- терактивных, использующих экран процессов при задании им неверных параметров обычно не происходит. Вместо этого появляется диалоговое окно с просьбой о повтор- ной попытке ввода параметров.

-возникновения фатальной ошибки (принудительно);

Третья причина завершения — ошибка, вызванная самим процессом, чаще всего связанная с ошибкой в программе. В качестве примеров можно привести неверную инструкцию, ссылку на несуществующий адрес памяти или деление на нуль. В не- которых системах (например, UNIX) процесс может сообщить операционной системе о своем намерении обработать конкретные ошибки самостоятельно, в таком случае, когда встречается одна из таких ошибок, процесс получает сигнал (прерывается), а не завершается.

-уничтожения другим процессом (принудительно).

Четвертая причина, из-за которой процесс может быть завершен, — это выполнение процессом системного вызова, приказывающего операционной системе завершить некоторые другие процессы. В UNIX этот вызов называется *kill*. Соответствующая функция Win32 называется *TerminateProcess*. В обоих случаях у процесса, вызывающего завершение, должны быть на это соответствующие полномочия. В некоторых системах при добровольном или принудительном завершении процесса тут же завершаются и все созданные им процессы. Но ни UNIX, ни Windows так не делают.

1. СОСТОЯНИЯ

Существует три состояния в которых может находится процесс

выполняемый (в данный момент использующий центральный процессор);

готовый (работоспособный, но временно приостановленный, чтобы дать возмож- ность выполнения другому процессу);

заблокированный (неспособный выполняться, пока не возникнет какое-нибудь

внешнее событие).

Логически первые два состояния похожи друг на друга. В обоих случаях процесс желает выполняться, но во втором состоянии временно отсутствует доступный для этого процессор. Третье состояние коренным образом отличается от первых двух тем, что процесс не может выполняться, даже если процессору кроме него больше нечем заняться.

