Семинар 3. Процессы в Unix. Основы работы с файлами.

На предыдущем семинаре рассматривалась организация процессов в Unix. Были написаны простейшие программы, создающие дочерний процесс с помощью функции fork(). Так же были даны общие сведения о том, что можно делать с созданным процессом. На этом семинаре подробно разберем запуск сторонних программ в созданном процессе с помощью функций exec().

Завершение процесса. Функция exit(). Существует два способа корректного завершения процесса в программах, написанных на языке С. Первый способ мы использовали до сих пор: процесс корректно завершался по достижении конца функции main() или при выполнении оператора return в функции main(), второй способ применяется при необходимости завершить процесс в каком-либо другом месте программы. Для этого применяется функция exit() из стандартной библиотеки функций для языка С. При выполнении этой функции происходит сброс всех частично заполненных буферов ввода-вывода с закрытием соответствующих потоков, после чего инициируется системный вызов прекращения работы процесса и перевода его в состояние закончил исполнение.

Возврата из функции в текущий процесс не происходит и функция ничего не возвращает.

Значение параметра функции exit() – кода завершения процесса – передается ядру операционной системы и может быть затем получено процессом, породившим завершившийся процесс. На самом деле при достижении конца функции main() также неявно вызывается эта функция со значением параметра 0.

Если процесс завершает свою работу раньше, чем его родитель, и родитель явно не указал, что он не хочет получать информацию о статусе завершения порожденного процесса, то завершившийся процесс не исчезает из системы окончательно, а остается в состоянии закончил исполнение либо до завершения процесса-родителя, либо до того момента, когда родитель соблаговолит получить эту информацию. Процессы, находящиеся в состоянии закончил исполнение, в операционной системе принято называть процессами-зомби (zombie, defunct).

Семейство функций *exec().* Для изменения пользовательского контекста процесса применяется системный вызов *exec()*, который пользователь не может вызвать непосредственно. Вызов *exec()* заменяет пользовательский контекст текущего процесса на содержимое некоторого исполняемого файла и устанавливает начальные значения регистров процессора. Этот вызов требует для своей работы задания имени исполняемого файла, аргументов командной строки и параметров окружающей среды. Для осуществления вызова программист может воспользоваться одной из шести функций:

execlp(), execvp(), execl(), execv(), execle(), execve(), отличающихся друг от друга представлением параметров, необходимых для работы системного вызова *exec()*.

Прототипы функций:

#include <unistd.h>

*int execlp(const char *file, const char *arg0,… const char *argN, (char *)NULL);*

*int execvp(const char *file, char *argv[]);*

*int execl(const char *path, const char *arg0,… const char *argN, (char *)NULL);*

*int execv(const char *path, char *argv[]);*

*int execle(const char *path, const char *arg0,… const char *argN, (char *)NULL, char *envp[]);*

*int execve(const char *path, char *argv[], char *envp[]);*

file — указатель на имя файла, который должен быть загружен.

path — указатель на полный путь к файлу, который должен быть загружен.

arg0,… ,argN — указатели на аргументы командной строки.

argv — массив из указателей на аргументы командной строки.

envp — массив указателей на параметры окружающей среды.

Суффиксы *l, v, p* и *e*, добавляемые к имени семейства *exec()* обозначают, что данная функция будет работать с некоторыми особенностями:

p — определяет, что функция будет искать «дочернюю» программу в директориях, определяемых переменной среды DOS PATH. Без суффикса *p* поиск будет производиться только в рабочем каталоге. Если параметр *path* не содержит маршрута, то поиск производится в текущей директории, а затем по маршрутам, определяемым переменной окружения PATH.

l — показывает, что адресные указатели (*arg0, arg1,..., argn*) передаются, как отдельные аргументы. Обычно суффикс *l* употребляется, когда число передаваемых аргументов заранее вам известно.

v — показывает, что адресные указатели (*arg[0], arg[1],...arg[n]*) передаются, как массив указателей. Обычно, суффикс *v* используется, когда передается переменное число аргументов.

e — показывает, что «дочернему» процессу может быть передан аргумент *envp*, который позволяет выбирать среду «дочернего» процесса. Без суффикса *e* «дочерний» процесс унаследует среду «родительского» процесса.

В случае успешного выполнения возврата из функций в программу, осуществившую вызов, не происходит, а управление передается загруженной программе.

В случае неудачного выполнения в программу, инициировавшую вызов, возвращается отрицательное значение.

Использование функций *execlp(…), execvp(…), execl(…), execv(…).* Написать программы: 1) Запуск системной программы из аргументов командной строки (*execvp*), 2) Запуск системной программы с заданием в тексте программы (*execlp*), 3) Запуск пользовательской программ из аргументов командной строки (*execv*), 4) Запуск пользовательской программы с заданием в тексте программы (*execl*).

Переменные окружения. Окружение (или среда) – это набор пар ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ, доступных каждому процессу. Для того, чтобы посмотреть окружение нужно ввести команду env. Переменные окружения могут формироваться как из заглавных, так и из строчных символов, однако исторически сложилось именовать их в верхнем регистре. Мы также не будем отступать от этого неписанного правила.

Про полезность окружения можно говорить долго, но основное его назначение - заставить одни и те же программы работать у разных пользователей по-разному. Приятно, например, когда программа "угадывает" имя пользователя или домашний каталог пользователя. Чаще всего такая информация "добывается" из переменных окружения USER и HOME соответственно.

Значение каждой переменной окружения изначально представляет собой строковую константу (строку). Интерпретация значений переменных полностью возлагается на программу. Иными словами, все переменные окружения имеют тип *char**, а само окружение имеет тип *char***. Чтобы вывести на экран значение какой-нибудь переменной окружения, достаточно набрать *echo $ИМЯ_ПЕРЕМЕННОЙ*.

Вообще говоря, при работе с оболочкой bash, запись $ИМЯ_ПЕРЕМЕННОЙ заменяется на само значение переменной, если только эта запись не встречается в кавычках, апострофах или в комментариях. В моем случае, например, запись $HOME заменяется на /home/nn. То есть команда mkdir $HOME/mynewdir создаст в моем домашнем каталоге подкаталог mynewdir.

В разных системах и у разных пользователей окружение отличается не только значениями переменных, но и наличием/отсутствием этих переменных. Ниже приведены те переменные окружения, которые есть почти у всех пользователей Linux:

USER - имя текущего пользователя

HOME - путь к домашнему каталогу текущего пользователя

PATH - список каталогов, разделенных двоеточиями, в которых производится "поиск" программ

PWD - текущий каталог

OLDPWD - предыдущий текущий каталог

TERM - тип терминала

SHELL - текущая командная оболочка

Некоторые переменные окружения имеются не во всех системах, но все-таки требуют упоминания:

HOSTNAME - имя машины

QTDIR - расположение библиотеки QT

MAIL - почтовый ящик

LD_LIBRARY_PATH - место "поиска" дополнительных библиотек

MANPATH - место поиска файлов man-страниц (каталоги, разделенные двоеточием)

LANG - язык и кодировка пользователя (иногда LANGUAGE)

DISPLAY - текущий дисплей в X11

Помимо переменных окружения, командные оболочки, такие как bash располагают собственным набором пар ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ. Это переменные оболочки. Набор таких переменных называют окружением (или средой) оболочки. Эти переменные чем-то напоминают локальные (стековые) переменные в языке С. Они недоступны для других программ (в том числе и для env) и используются в основном в сценариях оболочки. Чтобы задать переменную оболочки, достаточно написать в командной строке ПЕРЕМЕННАЯ=ЗНАЧЕНИЕ. Однако, при желании, можно включить локальную переменную оболочки в основное окружение. Для этого используется команда export ПЕРЕМЕННАЯ.

Загрузка окружения в программу. Для загрузки окружения в программу добавим в функцию *main()* еще один параметр:

*int main(int argc, char *argv[], char *envp[]).*

Параметр *envp* является массивом указателей на параметры окружающей среды процесса.

Задание: написать программу для вывода значений всех переменных окружения.

Использование функций *execle(…), execve(…).* Написать программу для запуска в дочернем процессе команды отображения содержимого домашней директории (*ls $HOME*).