ОП.14 **Основы функционирования UNIX - систем**

ЗАНЯТИЕ 10

Программное управление учетными записями

Управление учетными записями пользователей в операционной системе UNIX возможно как с помощью инструментов командной строки, так и с использованием интерфейса API («эй-пи-ай») или библиотечных функций языка C («си»).

В этих случаях пользователи могут реализовать свои, довольно сложные и изощренные алгоритмы управления учетными записями пользователей.

Также пользователи могут получать различного рода информацию, касающуюся учетных записей.

Программное управление учетными записями

В языке C («си») определен набор библиотечных функций, которые не имеют прямого соответствия в интерфейсах прикладного программирования (API), но, тем не менее, довольно широко используются при разработке программ для UNIX.

Особое внимание стоит обратить на то, что в C («си») имеется целый ряд библиотечных функций, не определенных в стандарте C («си»).

Тем не менее, эти библиотечные функции C («си») доступны во всех UNIX-системах.

Программное управление учетными записями

В примерах программ мы будем использовать как системные вызовы UNIX, так и библиотечные функции C («си»).

Программы, представленные в этом разделе, будут работать во всех наиболее популярных операционных системах (Linux, FreeBSD, Solaris и т. д.).

Компиляцию исходных текстов программ можно выполнить стандартными средствами, входящими в состав UNIX.

Программное управление учетными записями

Например, в операционной системе Linux, равно как и в других системах, можно использовать популярный пакет g++ компилятора C («си») со стандартными опциями.

Следующий фрагмент программного кода позволяет отобразить на экране дисплея **домашний каталог пользователя**, регистрационное имя которого указано в качестве аргумента программы.

Исходный текст программы представлен в листинге программы на следующей странице.

Вывод имени домашнего каталога пользователя

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <pwd.h>

int main(int argc, char* argv[])
{
 struct passwd *pwd;
 if  (argc != 2)
 {
  printf("Usage: %s registration name\n",
  argv[1]); exit(0);
 }
```

Вывод имени домашнего каталога пользователя

```
  pwd = getpwnam(argv[1]);
  if (!pwd)
  {
   printf("%s is not a valid user name!\n", argv[1]);
   exit(1);
  }
 printf("Home directory for registration name %s is %s\n",
argv[1], pwd->pw dir);
 return 0;
}
```

Программное управление учетными записями

В заголовочном файле `<pwd.h>` определяется набор функций, предназначенных для получения информации об учетной записи пользователя.

Эта информация содержится в файле `/etc/passwd`, и, кроме того, здесь же определена структура `passwd`.

В поля этой структуры помещается информация из файла `/etc/passwd`.

Структура `passwd` имеет формат, представленный на следующей странице:

Программное управление учетными записями

```
struct passwd
 {
  char*    pw_name
  char*    pw_passwd
  int      pw_uid
  int      pw_gid
  char*    pw_age
  char*    pw_comment
  char*    pw_dir
  char*    pw_shell
}
```

Программное управление учетными записями

Поля структуры имеют такой смысл:

- `pw_name` — регистрационное имя пользователя;
- `pw_passwd` — зашифрованный пароль;
- `pw_uid` — идентификатор пользователя;
- `pw_gid` — идентификатор (`gid`) группы;
- `pw_age` — минимальный срок действия пароля;
- `pw_comment` — общая информация о пользователе;
- `pw_dir` — начальный каталог пользователя;
- `pw_shell` — регистрационный командный интерпретатор `shell`.

Программное управление учетными записями

В данном примере используется функция `getpwnam()`, имеющая синтаксис:

```
const struct passwd* getpwnam(const char* имя_пользователя)
```

Здесь *имя_пользователя* — регистрационное имя пользователя.

Функция заполняет поля структуры `passwd` информацией о данной учетной записи.

Программное управление учетными записями

В другом примере приведен исходный текст программы, которая выводит на экран дисплея **регистрационное имя пользователя** и **путь к регистрационному командному интерпретатору**.

Имя и путь выводятся при заданном значении идентификатора `uid` пользователя, который является единственным параметром программы.

Листинг программы приведён на следующей странице.

Вывод имени пользователя и пути к интерпретатору shell

```
int main(int argc, char* argv[])
{
 struct passwd *pwd;
 int uid;

 if  (argc  != 2)
 {
  printf("Usage: %s uid\n", argv[0]);
  exit(0);
 }
 uid = atoi(argv[1]);
```

Вывод имени пользователя и пути к интерпретатору shell

```
 pwd = getpwuid(uid);


 if (!pwd)
 {
  printf("%s is not a valid user UID!\n",argv[l]);
  exit(1);
 }
 printf("Registration shell for user with uid = %d is: %s\n",
pwd->pw uid, pwd->pw shell);
 printf("USER_NUME for user with UID = Id is: %s\n", pwd->pw
uid, pwd->pw name);
 return 0;
}
```

Программное управление учетными записями

В этой программе используется функция `getpwuid ()`, синтаксис которой таков:

```
const struct passwd* getpwuid(const int uid)
```

В качестве параметра функция принимает значение идентификатора пользователя *uid*.

Программа, исходный текст которой показан на следующем листинге, выводит на экран дисплея имена и идентификаторы пользователей, записи о которых находятся в файле `/etc/passwd`.

Вывод регистрационных имен и идентификаторов всех пользователей

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <pwd.h>

int main(void)
{
 struct passwd *pwd;
 setpwent();
```

Вывод регистрационных имен и идентификаторов всех пользователей

```
 while (pwd = getpwent())
   {
    printf("Registration name:%s, uid:%d\n", pwd->pw name,
pwd->pw uid);
   }
 endpwent();
 return 0;
}
```

Программное управление учетными записями

В этой программе используются функции `setpwent()`, `getpwent()` и `endpwent()`.

Функция `setpwent ()` устанавливает указатель чтения на начало файла `/etc/passwd`, функция `getpwent()` смещает указатель на следующую запись файла `/etc/passwd`, а функция `endpwent()` закрывает файл `/etc/passwd`.

Для отображения идентификатора группы, зная ее имя, можно воспользоваться программой, исходный текст которой представлен на следующем листинге. Программа принимает единственный параметр, которым является имя группы.

Отображение идентификатора группы

```
#include <stdio.h>
#include <stdlib.h>
#include <grp.h>


int main(int argc, char* argv[])
{
 struct group *grp;
 if  (argc != 2)
 {
  printf("Usage: %s [group name]\n", argv[0]);
```

Отображение идентификатора группы

```
  exit(0);
 }
 grp = getgrnam(argv[1]);
 printf("gid = %d for group %s\n", grp->gr gid, grp->gr name);
 return 0;
}
```

Программное управление учетными записями

В файле заголовка `<grp.h>` определяется набор функций, предназначенных для получения информации о группах.

Эта информация содержится в файле `/etc/group`.

Кроме того, здесь же определена структура `group`, в поля которой помещается информация из файла `/etc/group`.

Структура `group` имеет формат, представленный на следующей странице:

Программное управление учетными записями

```
struct group
  {
    char*      gr_name
    char*      gr_passwd
    int        gr_gid
    char*      pw_comment
}
```

Программное управление учетными записями

Здесь:

- `gr_name` — имя группы;
- `gr_passwd` — зашифрованный пароль группы;
- `gr_gid` — идентификатор группы `gid`;
- `pw_comment` — имена членов группы.

Функция `getgrnam()` принимает в качестве аргумента имя группы и возвращает указатель на запись типа `struct group`, которая содержит информацию о группе, если группа определена в системе.

Программное управление учетными записями

В последнем примере, представленном на следующем листинге, показан исходный текст программы, которая отображает на экране дисплея:

- имя группы;
- ее идентификатор.

Эта информация содержится в файле `/etc/group`.

Вывод имени группы и ее идентификатора на экран

```
#include <stdio.h>
#include <stdlib.h>
#include <grp.h>

int main(void)
{
 struct group *grp;
 setgrent();
```

Вывод имени группы и ее идентификатора на экран

```
 while  (grp = getgrent())
 {
  printf("Group name : %s, gid: %d\n", grp->gr name, grp->gr
gid);
 }
 endgrent();
 return 0;
}
```

Программное управление учетными записями

В этой программе определённые функции выполняют следующие действия.

Функция `setgrent()` устанавливает указатель чтения файла на начало файла `/etc/group`.

Функция `getgrent()` смещает на следующую запись файла `/etc/group`.

Функция `endgrent()` закрывает файл `/etc/group`.

Программное управление учетными записями

Рассмотренные примеры демонстрируют только небольшую часть тех возможностей, которые предоставляет операционная система для создания собственных алгоритмов управления учетными записями пользователей и получения различного рода информации о пользователях.

В состав библиотечных функций С («си») включена группа функций, предназначенных для шифрования и дешифрования данных.

Функции этой группы очень важны, поскольку позволяют обеспечить безопасность системы.

Программное управление учетными записями

Так, например, файлы пользовательских паролей и системных данных, которым необходима высокая степень защиты, обычно должны храниться в зашифрованном виде.

Файл заголовка `crypt.h` определяет несколько функций шифрования/дешифрования — `crypt()`, `setkey()` и `encrypt()`.

Функция `crypt()` используется в UNIX-системах для шифрования пользовательских паролей и проверки действительности пароля пользователя.

Программное управление учетными записями

Функции `setkey()` и `encrypt()` выполняют действия, аналогичные тем, которые выполняет функция `crypt()`.

Разница лишь в том, что в этих функциях используется алгоритм шифрования данных по другому стандарту (DES).

Этот стандарт более надежен, чем тот, что используется функцией `crypt()`.

Вспомним некоторые команды

При использовании команды `ls -l`, для файла `/bin/sh`:

```
$ ls -l /bin/sh
```

был получен следующий результат:

```
-rwxr-x--x 1 root  bin  87924 Sep 21 2005 /bin/sh
```

Проанализируем полученный результат.

Здесь в первом поле задается тип файла и маска режима доступа к нему: поскольку первым символом является **дефис**, то это обычный файл.

Вспомним некоторые команды
Обозначение типов файлов в выводе команды
`ls -l`

Тип файла	**Символ**	**Создается командой**	**Удаляется командой**
Обычный файл	—	Редакторы, `cp` и др.	—
Каталог	`d`	`mkdir`	`rmdir`, `rm -r`
Файл байт-ориенти-рованного устройства	`c`	`mknod`	`rm`

Вспомним некоторые команды

Следующие за дефисом девять символов первого поля представляют триады битов режима, обозначенные литерами `r`, `w` и `x` (чтение, запись и выполнение соответственно).

Для данного примера **владелец** обладает полным доступом к файлу (`rwx`).

Пользователи группы `bin` — правом на чтение и выполнение (`r-x`).

Остальные пользователи могут только выполнить этот файл (`--x`).

Вспомним некоторые команды

Для копирования одного каталога в другой можно выполнить команду:

```
# cp -r  DIR  DIR.OLD
```

Здесь каталог `DIR` вместе со своим содержимым копируется в каталог `DIR.OLD`.

```
# cp -r  DIR1  DIR2  DIR.OLD
```

Здесь каталог `DIR1` и `DIR2` вместе со своим содержимым копируется в каталог `DIR.OLD`.

Вспомним некоторые команды

Перемещение файлов в операционной системе UNIX выполняется с помощью команды `mv`, имеющей синтаксис:

`mv [`*`опции...`*`]  `*`исходный_файл  файл_назначения`*

`mv [`*`опции...`*`]  `*`исходный_файл...  каталог`*

Вот пример использования команды `mv`:

```
# mv test test.old
```

Здесь файл `test` переименовывается в файл `test.old`.

Вспомним некоторые команды

Если последний параметр команды указывает на имя существующего каталога, то `mv` перемещает указанные файлы в этот каталог.

В том случае, если в качестве параметров заданы имена двух файлов, то имя первого файла будет изменено на имя второго.

Если же последний параметр не является каталогом, и за даны имена более чем двух файлов, то команда генерирует ошибку.

Контрольные вопросы

1. При выполнении команды `ls -l` для файла `/usr/data`:

```
$ ls -l /usr/data
```

был получен следующий результат:

```
drw-r----x 1 user1 usr 32544 Dec 16 2015 /usr/data
```

Что может означать эта запись?

Контрольные вопросы

1. При выполнении команды `ls -l` для файла `/usr/data`:

```
$ ls -l /usr/data
```

был получен следующий результат:

```
-rwxr-xr-- 2 user1 user2 usr 62234 Dec 02 2018
/usr/data
```

Что может означать эта запись?

Контрольные вопросы

1. При выполнении команды `ls -l` для файла `/usr/data/text1`:

```
$ ls -l /usr/data/text1
```

был получен следующий результат:

```
--w-r-----x 3 user1 user2 user3 usr 56128 Dec 28
2014 /usr/data/text1
```

Что может означать эта запись?

Контрольные вопросы

2. Выполните команду копирования каталога `BUH` в каталог `USR5`.

2. Выполните команду копирования каталога `SLV` в каталог `DIR`.

2. Выполните команду копирования каталога `DIR2` в каталог `ALX`.

2. Выполните команду копирования каталога `USL` и `RTX` в каталог `DIR3`.

2. Выполните команду копирования каталога `UVS` и `RST` в каталог `SLC`.

2. Выполните команду копирования каталога `CTS` и `DIR3` в каталог `SLV2`.

Контрольные вопросы

3. Переименовать файл `data2` в файл `old12`.
3. Переименовать файл `text0` в файл `text3`.
3. Переименовать файл `adata` в файл `old01`.
3. Переименовать файл `data3` в файл `test`.
3. Переименовать файл `test6` в файл `old`.
3. Переименовать файл `data2` в файл `old5`.

Контрольные вопросы

4. Переместить файл `data` в каталог `DIR2`.
4. Переместить файл `text` в каталог `BUH`.
4. Переместить файл `test` в каталог `OLD`.
4. Переместить файл `data` в каталог `USR`.
4. Переместить файл `text2` в каталог `LIB`.
4. Переместить файл `text4` в каталог `DIR`.

Список литературы:

1. Юрий Магда. UNIX для студентов, Санкт-Петербург «БХВ-Петербург», 2007.
2. Unix и Linux: руководство системного администратора, 4-е издание, 2012, Э. Немет, Г. Снайдер, Т. Хейн, Б. Уэйли
3. Организация UNIX систем и ОС Solaris 9, Торчинский Ф.И., Ильин Е.С., 2-е издание, исправленное, 2016.

Спасибо за внимание!

Преподаватель: Солодухин Андрей Геннадьевич

Электронная почта: asoloduhin@kait20.ru