Systems and Network Management

Assignment 3: Creating bulk posix accounts in an LDAP Directory

Submission: by 8pm, Sunday, 23 May 2004

Where: Online at `http://nicku.org/perl2/submit.cgi`. A paper submission is not required (save the trees!)

Important: All programs you write must start with these lines:

```
#! /usr/bin/perl

use warnings;
use strict;
```

and should compile without errors.

Cheating: Your work *must* be original. Copying will be *severely* dealt with.

1 Background

This assignment requires the use of the `Net::LDAP` Perl module. There is plenty of documentation that comes with the software.

I have made a unique "student registration system file" for each student in the class. Please make sure that you use the correct one as your input data, or you will receive *zero marks* for the output from your program.

You will create several hundred LDAP POSIX accounts from this student registration system file.

Feel free to reuse the code you wrote for your first Perl account creation assignment, since the format of the data has not changed significantly.

You may wish to use the standard `Getopt::Long` Perl module as a way to pass options to your software. See `perldoc Getopt::Long`.

Assignment Requirements

1. Write a program that can read the data provided at `http://ictlab.tyict.vtc.edu.hk/snm/assignments/assignment-perl-ldap/data/artificial-student-data-`⟨*Your student ID*⟩⟨*Your .txt*⟩, where ⟨*Your student ID*⟩ is your nine-digit student number.

   Your program will generate LDIF to create a POSIX LDAP account for each user, and create a single private user group for each user. Make each user a primary member of their own private user group.

2. As an option, your program should be able to generate accounts directly on the directory server. In this case, your software should check whether the account or group entry exists before creating it, and should inform the user about all error conditions.

Note: I no longer recommend that you use the built in Perl functions `getpwent` and `getgrent` to determine the next available unused `uidNumber` and `gidNumber`. I suggest that instead you either:

- simply start the `uidNumber` and `gidNumber` from 2000 if your directory has no accounts, since these numbers are only relevant in your own directory.

OR:

- use the method that I outlined in an email I sent you earlier, where you search for the highest `uidNumber` and `gidNumber` values in your directory, and start with a number 1 greater than that, or with 2000 if there are no accounts in your directory.

3. The root of your directory shall be `ou=`⟨*Your Student ID*⟩`,o=ICT`. The user accounts shall be under `ou=People,ou=`⟨*Your Student ID*⟩`,o=ICT`. The groups shall be under `ou=Group,ou=`⟨*Your Student ID*⟩`,o=ICT`.

4. Each user account entry shall be a member of the following object classes: `posixAccount`, `shadowAccount` and any suitable `structural` class such as `inetOrgPerson`.

5. Each group entry should be a member of the `posixGroup` object class, and should contain an attribute `cn` which is equal to the user ID for whom this is a private group. It should also contain an attribute `gidNumber` with a unique integer number greater than 1999 that has the same value as the `gidNumber` attribute that is in the corresponding user entry.

6. Each user account should have an attribute `loginShell` with the default value `/bin/bash`, and an attribute `homeDirectory` with the value `/home/`⟨*user name*⟩. Each user entry should also have a `gidNumber` attribute (as mentioned above), and a `uidNumber` attribute. Both the `uidNumber` and `gidNumber` attributes should have unique values over 1999.

7. The program should also provide an option to generate LDIF to delete the accounts read from the file(s) in the input.

8. The program does not need to create a home directory for each user.

9. The username should be the student number of the student and shall be stored in the `uid` attribute.

> For example, for a student with the name "YEUNG, Hoi Man," and the student number 915367894, the user name that the person logs in with is `915367894`.

10. The password shall be their Hong Kong ID, with letters in *lower* case, but `not` including the parentheses or the check digit. So for a student with the Kong Kong ID A456789(A), their password should be `a456789`. This means that there will be seven characters in each password.

You are to create a `userPassword` attribute in each account, putting into it the text which is the MD5 hash of the password. You can create this using either the `unix_md5_crypt()` method of the `Crypt::PasswdMD5` module you can install from CPAN, or use the `hash_md5_password()` function you used in your previous assignment.

The `userPassword` attribute should begin with the string `{crypt}`. So for example, if a student number is A123456(3), then the output of:

```
use Crypt::PasswdMD5;
our $clear_passwd = shift || 'a123456';
our $hashed_password = unix_md5_crypt $clear_passwd;
print "{crypt}$hashed_password\n";
```

may look like this:

```
 $ ./make-password.pl
{crypt}$1$xM8WAHwg$qwA3/nAhJRSSQ75qM6IdT0
```

and be suitable as a value for the `userPassword` attribute.

11. You should include the full name of each student in their `cn` attribute, their family name will be stored in the `sn` attribute, their given name will be stored in the `givenName` attribute.

12. More marks will be awarded for solutions that are more general, more flexible and of course, better structured.

13. Also, with the soft copy of your program, provide a soft copy of the following:

- an LDIF file output from your program that creates all the accounts and creates the private user groups for all the students in the artificial student data that was allocated to you.
- an LDIF file output from your program that deletes all the accounts and private user groups listed in the artificial student data file that was allocated to you.

14. Your program should display an informative error message for each account that fails to be created or deleted.

15. You should be prepared to demonstrate your program in the laboratory.

If you have any questions about the assignment, please ask them by email. I will send the reply to all students if that seems helpful, but I will conceal the identity of the person who asked the question. I welcome any questions.