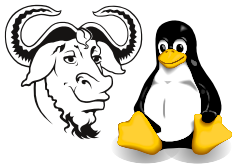

Using Regular Expressions to Create User Accounts — Solutions

For each exercise where you write a program, keep the original program from each exercise and modify a copy for the next exercise.
Submission of assignment: You will need to submit all your programs by email to me at `nicku@vtc.edu.hk`. See the subject web site for the format of the subject line of your assignments.
Due: The deadline for submission is 12.30 pm on the Tuesday of the following week.

1. Download the artificial student data from `http://nicku.org/snm/lab/regular-expressions/artificial-student-data.txt`. This file is in the old format of the student registration system, but contains no real data about any student.

2. Write a Perl program that can read all the lines of this file when it is given as a command line parameter, and print the student numbers only, one to each line.

```
#! /usr/bin/perl -w

while ( <> )
{
    print "$1\n" if /(\d{9})/;
}
```

3. Make a copy of your program, and modify it so that it prints only the names of the students, one to each line, with no extra spaces either at the beginning or end of each name.

```
#! /usr/bin/perl -w

while ( <> )
{
    print "$1\n" if /^.{6}(.*[^ ])  +[MF]  \d{9} /;
}
```

4. Using the manual page for the `useradd` program as a guide, modify your previous programs so that your program prints one `useradd` command for each student, using the student ID as the login ID, the Hong Kong ID as the password, and the name of the student as a comment.

```
#! /usr/bin/perl -w

use strict;
```

```
# Note: I provided this subroutine on the subject website
# I did not intend that you should need to write it;
# I simply forgot that useradd requires a hashed password, not
# a plain password.
sub hash_md5_password($)
{
    my $clear_text_password = shift;
    my $salt = join '',
    ('.', '/', 0..9, 'A'..'Z', 'a'..'z')[rand 64, rand 64, rand 64, rand 64,
                                         rand 64, rand 64, rand 64, rand 64];
    $salt = '$1$'.$salt.'$';
    my $hashed_password = crypt( $clear_text_password, $salt );
    return $hashed_password;
}

while ( <> )
{
    if ( /^.{6}(.*[^ ])  +[MF]  (\d{9})  ([a-zA-Z]\d{6}\([\dA-Z]\))/ )
    {
        my ( $name, $id, $hkid ) = ( $1, $2, $3 );
        my $hashed_passwd = hash_md5_password( $hkid );
        print "useradd -c \"$name\" -p $hashed_passwd $id\n";
    }
}
```

5. Modify this last program further so that it uses the built-in Perl function `system` to execute the `useradd` command as well as print the command. Type `perldoc -f system` to read about this important function.

```
#! /usr/bin/perl -w

use strict;

sub hash_md5_password($)
{
    my $clear_text_password = shift;
    my $salt = join '',
    ('.', '/', 0..9, 'A'..'Z', 'a'..'z')[rand 64, rand 64, rand 64, rand 64,
                                         rand 64, rand 64, rand 64, rand 64];
    $salt = '$1$'.$salt.'$';
    my $hashed_password = crypt( $clear_text_password, $salt );
    return $hashed_password;
}

while ( <> )
{
    if ( /^.{6}(.*[^ ])  +[MF]  (\d{9})  ([a-zA-Z]\d{6}\([\dA-Z]\))/ )
    {
        my ( $name, $id, $hkid ) = ( $1, $2, $3 );
        my $hashed_passwd = hash_md5_password( $hkid );
```

```
        my @cmd = (
                   'useradd',
                   '-c', "\"$name\"",
                   '-p', $hashed_passwd,
                   $id
                  );
        print "@cmd\n";
        system @cmd;
    }
}
```

6. Modify your last program so that it reports an error message if the execution of any `useradd` command is unsuccessful.

```
#! /usr/bin/perl -w

use strict;

sub hash_md5_password($)
{
    my $clear_text_password = shift;
    my $salt = join '',
    ('.', '/', 0..9, 'A'..'Z', 'a'..'z')[rand 64, rand 64, rand 64, rand 64,
                                         rand 64, rand 64, rand 64, rand 64];
    $salt = '$1$'.$salt.'$';
    my $hashed_password = crypt( $clear_text_password, $salt );
    return $hashed_password;
}

while ( <> )
{
    if ( /^.{6}(.*[^ ])  +[MF]  (\d{9})  ([a-zA-Z]\d{6}\([\dA-Z]\))/ )
    {
        my ( $name, $id, $hkid ) = ( $1, $2, $3 );
        my $hashed_passwd = hash_md5_password( $hkid );
        my @cmd = (
                   'useradd',
                   '-c', "\"$name\"",
                   '-p', $hashed_passwd,
                   $id
                  );
        print "@cmd\n";
        system( @cmd ) == 0 or warn( "@cmd failed\n" );
    }
}
```