|  |
| --- |
| Ex no : 2 |
| Question :  Create shell scripts for user management, system management, backup and restore processes. |
| **Create shell scripts for user management :**  You can quickly write a shell script that reads username, password from the keyboard, and add a username to [the /etc/passwd](https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/) and store encrypted password in [/etc/shadow file](https://www.cyberciti.biz/faq/understanding-etcshadow-file/) using useradd command.  The useradd command/adduser command used to create a new user on Linux and passwd command to set or change password for users.  Linux shell script to add a user with a password  Syntax :  useradd -m **-p EncryptedPasswordHere** username  Where,   * **-m** : The user’s home directory will be created if it does not exist. * **-p EncryptedPasswordHere** : The encrypted password, as returned by crypt(). * **username** : Add this user to the Linux system,   Step 1 – Create an encrypted password  You need to create an encrypted password using Perl [crypt()](https://perldoc.perl.org/functions/crypt.html) as follows:  crypt($plain, $salt)    *## perl one liner ##*  perl -e 'print crypt("Your-Clear-Text-Password-Here", "salt"),"\n"'  Please note that crypt() is a one-way hash function.  The PLAINTEXT ($plain) and SALT are turned into a short string, called a digest, which is returned.  The same PLAINTEXT and SALT will always return the same string, but there is no (known) way to get the original PLAINTEXT from the hash.  Small changes in the PLAINTEXT or SALT will result in large changes in the digest.  Let us try out perl example: perl -e 'print crypt("2IL@ove19Pizza4\_", "salt"),"\n"' |
| Sample Output :  sa.KT9zrGYeg2  The Perl command will display the encrypted password (sa.KT9zrGYeg2) on screen. The Perl crypt() function is a one way encryption method meaning, once a password has been encrypted, it cannot be decrypted. The password string is taken from the user and encrypted with the salt and displayed back on computer screen. We can store an encrypted password using the following syntax:  password="1YelloDog@"  pass=$(perl -e 'print crypt($ARGV[0], "password")' $password)  echo "$pass"  Sample outputs”  paU5t8Al/qf6M |
| Shell script :  #!/bin/bash  if [ $(id -u) -eq 0 ]; then  read -p "Enter username : " username  read -s -p "Enter password : " password  egrep "^$username" /etc/passwd >/dev/null  if [ $? -eq 0 ]; then  echo "$username exists!"  exit 1  else  pass=$(perl -e 'print crypt($ARGV[0], "password")' $password)  useradd -m -p "$pass" "$username"  [ $? -eq 0 ] && echo "User has been added to system!" || echo "Failed to add a user!"  fi  else  echo "Only root may add a user to the system."  exit 2  fi  Next set permissions using the chmod command: chmod +x add-user-script.sh  Run it as following $ **./add-user-script.sh**  $ sudo ./add-user-script.sh  Or run it as root user:  # ./adduser  Sample outputs:  Enter username : roja  Enter password : HIDDEN  User has been added to system!  Step 3 – Change existing Linux user’s password in one CLI  We are going use the chpasswd command that reads a list of user names and password pairs from the keyboard and uses this information to update a group of existing users.  The syntax is as follows:  echo "user\_name:password" | chpasswd  However, the passwords must be provided in clear-text format, and are encrypted by the chpasswd command. For example, set or change user password, run:  # echo 'vivek:@iLovePizzaEvery1day' | chpasswd  Verify that password has been changed using the chage command:  # chage -l vivek  We can use the grep command/egrep command to search for usernames:  grep "^username" /etc/passwd  grep "^tom" /etc/passwd  If the chpasswd command not installed, use your systems package manager tool such as apt command/apt-get command/dnf command/yum command to install the same. |
| Writing a script to display system information:  #!/bin/bash  # This script will return the following set of system information:  # -Hostname information:  echo -e "\e[31;43m\*\*\*\*\* HOSTNAME INFORMATION \*\*\*\*\*\e[0m"  hostnamectl  echo ""  # -File system disk space usage:  echo -e "\e[31;43m\*\*\*\*\* FILE SYSTEM DISK SPACE USAGE \*\*\*\*\*\e[0m"  df -h  echo ""  # -Free and used memory in the system:  echo -e "\e[31;43m \*\*\*\*\* FREE AND USED MEMORY \*\*\*\*\*\e[0m"  free  echo ""  # -System uptime and load:  echo -e "\e[31;43m\*\*\*\*\* SYSTEM UPTIME AND LOAD \*\*\*\*\*\e[0m"  uptime  echo ""  # -Logged-in users:  echo -e "\e[31;43m\*\*\*\*\* CURRENTLY LOGGED-IN USERS \*\*\*\*\*\e[0m"  who  echo ""  # -Top 5 processes as far as memory usage is concerned  echo -e "\e[31;43m\*\*\*\*\* TOP 5 MEMORY-CONSUMING PROCESSES \*\*\*\*\*\e[0m"  ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6  echo ""  echo -e "\e[1;32mDone.\e[0m" |
| Next, give the script execute permissions:  # chmod +x system\_info.sh  and run it:  ./system\_info.sh |