
Remote Calculator

Expect + Cron job + Email
notice

Presented by
Sai Harshinee Roopakula
19577

Table of Contents

- Introduction
 - SSH
 - Expect
 - Cron job
- SSH Setup
- Problem Statement
- Design
 - Creating an Interactive Calculator on the Server side
 - Shell script to execute the Interactive Calculator on client side.
 - Setting up a cron job to automate the process at 1 am every day.
 - Configuring the cron job to support email notice
 - Removing the cron job
- Conclusion
- Bibliography

Introduction - SSH

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

SSH provides a secure channel over an unsecured network by using a client-server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access Unix-like operating systems, but it can also be used on Microsoft Windows. Windows 10 uses OpenSSH as its default SSH client and SSH server.

SSH was designed as a replacement for Telnet and for unsecured remote shell protocols such as the Berkeley rsh and the related rlogin and rexec protocols. Those protocols send sensitive information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet.

Introduction - Expect

Expect command or expect scripting language is a language that talks with your interactive programs or scripts that require user interaction.

Expect scripting language works by expecting input, then the Expect script will send the response without any user interaction. You can say that this tool is your robot, which will automate your scripts.

Some of the Expect command which are used for interaction are:

- **spawn** - Starts a script or a program.
- **expect** - Waits for program output.
- **send** - Sends a reply to your program.
- **interact** - Allows you to interact with your program.

The spawn command starts a script or a program like the shell, FTP, Telnet, SSH, SCP, and so on.

The send command sends a reply to a script or a program.

The Expect command waits for input.

The interact command allows you to define a predefined user interaction.

Introduction - Cron job

The cron command-line utility, also known as **cron job** is a job scheduler on Unix-like operating systems.

Users who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals.

It typically automates system maintenance or administration—though its general-purpose nature makes it useful for things like downloading files from the Internet and downloading email at regular intervals.

Cron is most suitable for scheduling repetitive tasks. Scheduling one-time tasks can be accomplished using the associated at utility.

To schedule a cron job to run on May 23 10.30 am. This can be done with the following command:

```
30 10 23 06 * sh /root/sh
```

Here,

30 => 30th minute

10 => 10th hour

23 => 23rd day of month

06 => June

* => Every day of week

SSH Setup

Step 1: Generating the key pair

Command: `ssh-keygen -t rsa`

This creates a public/private keypair of the type (-t) rsa.

It prompts "Enter the file in which you wish to save they key (i.e., /home/username/.ssh/id_rsa)." Click on Enter.

Once the key pair is created, you are prompted to Enter the paraphrase (type anything of your choice). Click on Enter. It prompts to enter the Enter the same paraphrase again. Click on Enter.

```
[Hemanths-MacBook-Pro:~ hemanthharshinee$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/hemanthharshinee/.ssh/id_rsa):
/Users/hemanthharshinee/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/hemanthharshinee/.ssh/id_rsa
Your public key has been saved in /Users/hemanthharshinee/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:51KU74TuFe7aAOX+h1wwBaNV6S5Dpo+J1P4HCqoaGA hemanthharshinee@Hemanths-MacBook-Pro.local
The key's randomart image is:
+----[RSA 3072]-----+
|      .o oo      |
|      ..+oo      |
|      B.+        |
|      = o .       |
|      ..S.+       |
| .E   oB=0 .      |
| o .  =0=.+       |
| ... ..*O*+ o     |
|+.   ..Bo.o       |
+-----[SHA256]-----+
```

SSH Setup

Step 2: Copying the public key you just created on your home computer to the server

Command: `cat ~/.ssh/id_rsa.pub | ssh username@server.dreamhost.com "mkdir ~/.ssh; cat >> ~/.ssh/authorized_keys"`

This command assumes you do NOT already have an `/.ssh` directory under your server username. This command creates the `/.ssh` directory for you on your DreamHost server.

If you already have an `/.ssh` directory on your web server, type the following command:

Command: `cat ~/.ssh/id_rsa.pub | ssh username@server.dreamhost.com "cat >> ~/.ssh/authorized_keys"`

```
Hemanths-MacBook-Pro:~ hemanthharshinee$ cat ~/.ssh/id_rsa.pub | ssh 19577@35.167.127.201 "cat >> ~/.ssh/authorized_keys"
19577@35.167.127.201's password:
```

The commands above created a new folder under the server user named `/.ssh` with 755 permissions.

In that folder is your `authorized_keys` file which was just copied from your home computer which has 644 permissions.

SSH Setup

Login in to your server to update the permissions to further secure your keys.

Command: `ssh username@server`

```
[Hemanths-MacBook-Pro:~ hemanthharshinee$ ssh 19577@35.167.127.201
19577@35.167.127.201's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-1058-aws x86_64)

[
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

System information as of Mon Nov 1 22:21:03 PDT 2021

System load:  0.0                Processes:            99
Usage of /:   14.9% of 58.10GB    Users logged in:     0
Memory usage: 19%                IP address for eth0: 172.26.3.73
Swap usage:   0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Get cloud support with Ubuntu Advantage Cloud Guest:
  http://www.ubuntu.com/business/services/cloud

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

76 packages can be updated.
1 update is a security update.

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Nov 1 22:17:27 2021 from 24.6.105.240
```


SSH Setup

```
19577@CS522:~$ chmod 700 ~/.ssh
19577@CS522:~$ chmod 600 ~/.ssh/authorized_keys
```

Step 3: Confirming the SSH connection

If everything is configured properly, you should now be able to access your server account through SSH without a password. Run this command on your home computer where you just created the original keypair.

Command: `ssh username@server`

```
19577@CS522:~$ ssh 19577@35.167.127.201
The authenticity of host '35.167.127.201 (35.167.127.201)' can't be established.
ECDSA key fingerprint is SHA256:jGUnk2PP6ZhQ8NhJycmvJ/DJqikQiPAzmKyRoZrLuIw.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '35.167.127.201' (ECDSA) to the list of known hosts.
19577@35.167.127.201's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-1058-aws x86_64)
```

You should be able to login without password

SSH Setup

Step 4: Testing

Now you are in your local system.

Command: `ssh username@server ls`

```
[Hemanths-MacBook-Pro:~ hemanthharshinee$ ssh 19577@35.167.127.201 ls  
id_dsa.pub
```

Problem Statement

The problem statement is to:

Step 1: Create an Interactive Calculator on the Server side.

Step 2: Create shell script to automatically execute the server side's Interactive Calculator on the Client side.

Step 3: Set up a cron job to automate the process at 1 am every day.

Step 4: Configure the cron job to support email notice

Step 5: Remove the cron job after the project

Design - Interactive Calculator on Server Side

```
hemanthharshinee — ssh 19577@35.167.127.201 — 89x41
Hemanths-MacBook-Pro:~ hemanthharshinee$ ssh 19577@35.167.127.201
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-1058-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Nov  6 18:51:23 PDT 2021

System load:  0.01               Processes:            115
Usage of /:   15.0% of 58.10GB   Users logged in:     2
Memory usage: 28%               IP address for eth0: 172.26.3.73
Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

https://ubuntu.com/aws/pro

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

76 packages can be updated.
1 update is a security update.

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Nov  6 18:49:52 2021 from 24.6.105.240
19577@CS522:~$ vi myScript
```

```
hemanthharshinee — ssh 19577@35.167.127.201 — 89x41
echo -n "N1:"
read N1

echo -n "N2:"
read N2

echo -n "Operation(+, -, *, /):"
read Operation

if [ "$Operation" = "+" ]
then
    (( result = N1 + N2 ))
elif [ "$Operation" = "-" ]
then
    if [ "$N1" -ge "$N2" ]
    then
        (( result = N1 - N2 ))
    else
        (( result = N2 - N1 ))
        result="-${result}"
    fi
elif [ "$Operation" = "*" ]
then
    (( result = N1 * N2 ))
elif [ "$Operation" = "/" ]
then
    (( result1 = N1 / N2 ))
    (( result2 = N1 * 100 / N2 ))
    result="${result1}.${result2}"
else
    echo "Error: wrong operation $Operation"
    exit 1

fi

echo "Result:$result"
```

Design - Interactive Calculator on Server Side

myScript

```
echo -n "N1:"
read N1

echo -n "N2:"
read N2

echo -n "Operation(+, -, *, /):"
read Operation

if [ "$Operation" = "+" ]
then
  (( result = N1 + N2 ))
elif [ "$Operation" = "-" ]
then
  if [ "$N1" -ge "$N2" ]
  then
    (( result = N1 - N2 ))
  else
    (( result = N2 - N1 ))
  result="-${result}"
  fi
elif [ "$Operation" = "*" ]
then
  (( result = N1 * N2 ))
elif [ "$Operation" = "/" ]
then
  (( result1 = N1 / N2 ))
  (( result2 = N1 * 100 / N2 ))
  result="${result1}.${result2}"
else
  echo "Error: wrong operation $Operation"
  exit 1

fi

echo "Result:$result"
```

Output - myScript while running on Server side

```
hemanthharshinee — ssh 19577@35.167.127.201 — 78x33
* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

76 packages can be updated.
1 update is a security update.

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Nov  6 22:14:52 2021 from 24.6.105.240
19577@CS522:~$ ./myScript
[N1:3
[N2:4
[Operation(+, -, *, /):+
Result:7
19577@CS522:~$ ./myScript
[N1:4
[N2:10
[Operation(+, -, *, /):*
Result:40
19577@CS522:~$ ./myScript
[N1:4
[N2:23
[Operation(+, -, *, /):-
Result:-19

19577@CS522:~$ ./myScript
[N1:10
[N2:3
[Operation(+, -, *, /):/
Result:3.333
```

Design - Expect code on the Client side

```
Hemanths-MacBook-Pro:~ hemanthharshinee$ vi exp
#!/usr/bin/expect --

set timeout -1
set N1 [lindex $argv 0]
set N2 [lindex $argv 1]
set Operation [lindex $argv 2]
spawn ssh 19577@35.167.127.201 /home/19577/myScript
expect "N1:" { send "$N1\r" }
expect "N2:" { send "$N2\r" }
expect "Operation(+, -, *, /):" { send -- "$Operation\r" }
expect " " send { "exit\r" }
```

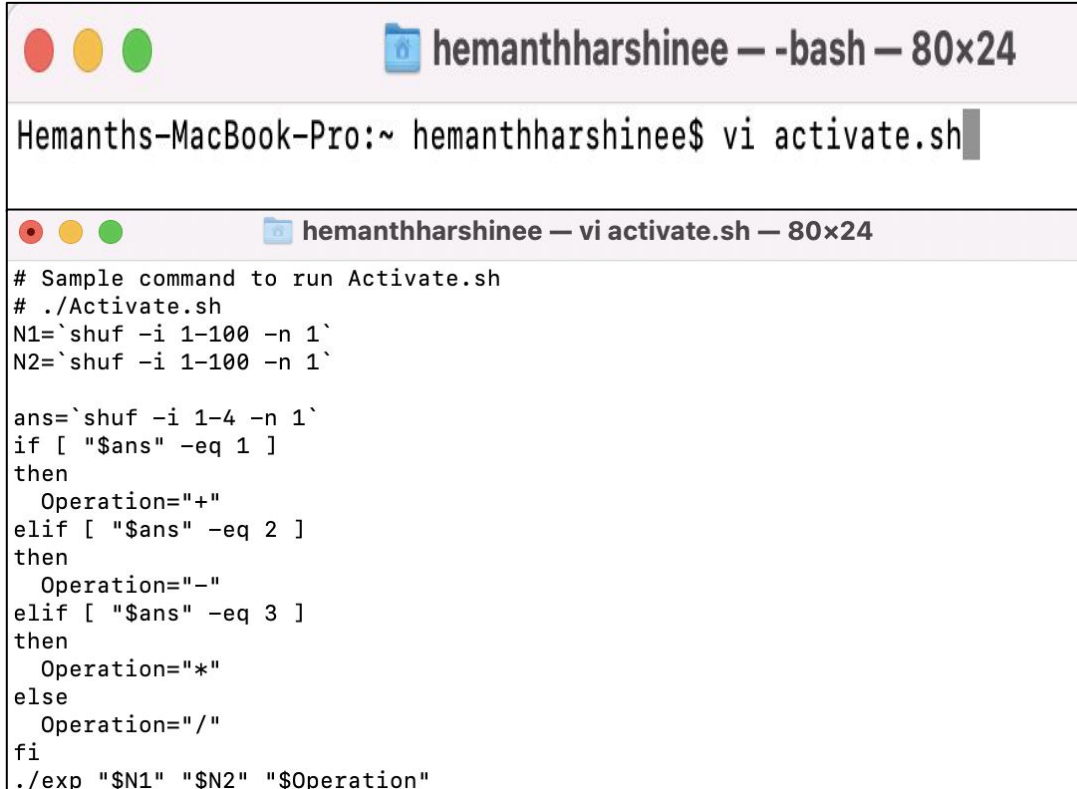
Design - Expect code on the Client side

exp

```
#!/usr/bin/expect --

set timeout -1
set N1 [lindex $argv 0]
set N2 [lindex $argv 1]
set Operation [lindex $argv 2]
spawn ssh 19577@35.167.127.201
/home/19577/myScript
expect "N1:" { send "$N1\r" }
expect "N2:" { send "$N2\r" }
expect "Operation(+, -, *, /):" { send
-- "$Operation\r" }
expect " " send { "exit\r" }
```


Design - Shell Script code to automatically execute the Server side code with expect



```
hemanthharshinee — -bash — 80x24
Hemanths-MacBook-Pro:~ hemanthharshinee$ vi activate.sh

hemanthharshinee — vi activate.sh — 80x24
# Sample command to run Activate.sh
# ./Activate.sh
N1=`shuf -i 1-100 -n 1`
N2=`shuf -i 1-100 -n 1`

ans=`shuf -i 1-4 -n 1`
if [ "$ans" -eq 1 ]
then
    Operation="+"
elif [ "$ans" -eq 2 ]
then
    Operation="-"
elif [ "$ans" -eq 3 ]
then
    Operation="*"
else
    Operation="/"
fi
./exp "$N1" "$N2" "$Operation"
```

Design - Shell Script code to automatically execute the Server side code with expect

activate.sh

```
# Sample command to run Activate.sh
# ./Activate.sh
N1=`shuf -i 1-100 -n 1`
N2=`shuf -i 1-100 -n 1`

ans=`shuf -i 1-4 -n 1`
if [ "$ans" -eq 1 ]
then
    Operation="+"
elif [ "$ans" -eq 2 ]
then
    Operation="-"
elif [ "$ans" -eq 3 ]
then
    Operation="*"
else
    Operation="/"
fi
./exp "$N1" "$N2" "$Operation"
```

Output - Shell script code on the Client side

```
Hemanths-MacBook-Pro:~ hemanthharshinee$ ./activate.sh
spawn ssh 19577@35.167.127.201 /home/19577/myScript
N1:87
N2:95
Operation(+, -, *, /):*
Result:8265
Hemanths-MacBook-Pro:~ hemanthharshinee$ ./activate.sh
spawn ssh 19577@35.167.127.201 /home/19577/myScript
N1:78
N2:40
Operation(+, -, *, /):-
Result:38
Hemanths-MacBook-Pro:~ hemanthharshinee$ ./activate.sh
spawn ssh 19577@35.167.127.201 /home/19577/myScript
N1:49
N2:99
Operation(+, -, *, /):+
Result:148
Hemanths-MacBook-Pro:~ hemanthharshinee$ ./activate.sh
spawn ssh 19577@35.167.127.201 /home/19577/myScript
N1:51
N2:61
Operation(+, -, *, /):*
Result:3111
Hemanths-MacBook-Pro:~ hemanthharshinee$ ./activate.sh
spawn ssh 19577@35.167.127.201 /home/19577/myScript
N1:88
N2:67
Operation(+, -, *, /):/
Result:1.131
```

Design - Setting up how to send email from localhost (Mac)

Step 1. Edit Postfix config file

Open a terminal and edit the file main.cf

Command: `sudo vi /etc/postfix/main.cf`

First check Postfix is configured correctly, look for the following lines (They are probably separated):

```
mail_owner = _postfix
setgid_group = _postdrop
sendmail_path = /usr/sbin/sendmail
newaliases_path = /usr/bin/newaliases
mailq_path = /usr/bin/mailq
setgid_group = _postdrop
html_directory = /usr/share/doc/packages/postfix/html
manpage_directory = /usr/share/man
sample_directory = /usr/share/doc/packages/postfix/samples
readme_directory = /usr/share/doc/packages/postfix/README_FILES
```

Design - Setting up how to send email from localhost (Mac)

Now add the following lines at the very end of the file:

```
#Gmail SMTP
relayhost=smtp.gmail.com:587
# Enable SASL authentication in the Postfix SMTP client.
smtp_sasl_auth_enable=yes
smtp_sasl_password_maps=hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options=noanonymous
smtp_sasl_mechanism_filter=plain
# Enable Transport Layer Security (TLS), i.e. SSL.
smtp_use_tls=yes
smtp_tls_security_level=encrypt
tls_random_source=dev:/dev/urandom
```

This is telling Postfix to use a GMAIL SMTP server with Simple Authentication and Security Layer (SASL) which will be stored in the path "/etc/postfix/sasl_passwd".

Design - Setting up how to send email from localhost (Mac)

Step 2. Creating the sasl_passwd file

We need to create the sasl_passwd file with the SMTP credentials

Command: `sudo vi /etc/postfix/sasl_passwd`

Write the following content and save:

```
smtp.gmail.com:587 your_email@gmail.com:your_password
```

Create the Postfix lookup table from the sasl_passwd file.

Command: `sudo postmap /etc/postfix/sasl_passwd`

This will create the file sasl_passwd.db

Step 3. Restart Postfix

To apply all new changes we have to restart Postfix:

Command: `sudo postfix reload`

Design - Setting up how to send email from localhost (Mac)

Step 4. Turn on less secure apps (Only Gmail)

In Gmail we must switch on the option "Access for less secure apps", otherwise we will get the error:
SASL authentication failed

Step 5. Test it!

Let's send a mail to our own account to be sure everything is working fine:

```
date | mail -s testing your_email@gmail.com
```

If we receive an email to the account we specified, the setup is completed.



Design - Setting up a cron job and sending email notice

First we need to specify the PATH as PATH of crontab is different from the path we are on. To know the path -

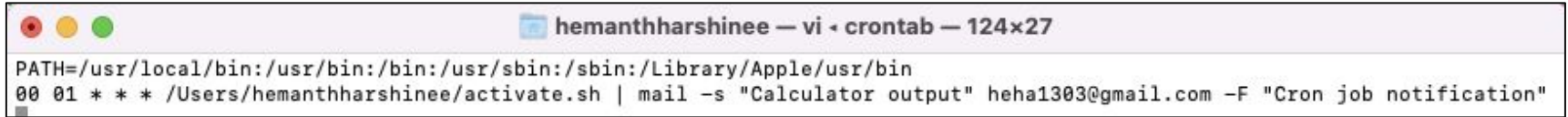
Command: echo \$PATH

To setup a Cron job - **Command** : crontab -e

As we need the Cron job to run at 1 am everyday the command is 00 10 * * * <mention complete path of the script>

We can then send that output to an email by using the mail command with necessary flags

-s is subject, -F is the Full name of the sender

A terminal window titled 'hemanthharshinee — vi • crontab — 124x27'. The window shows the output of the 'echo \$PATH' command as 'PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin'. Below this, the crontab file is being edited in vi, showing the line '00 01 * * * /Users/hemanthharshinee/activate.sh | mail -s "Calculator output" heha1303@gmail.com -F "Cron job notification"'.

```
hemanthharshinee — vi • crontab — 124x27
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
00 01 * * * /Users/hemanthharshinee/activate.sh | mail -s "Calculator output" heha1303@gmail.com -F "Cron job notification"
```

If you want to remove the cron jobs of currently logged in user, execute the following

Command: crontab -r

Output - Cron job email



Cron job notification

Calculator output

Inbox - Google 1:00 AM

[Details](#)

spawn ssh [19577@35.167.127.201](#) /home/19577/myScript

N1:89

N2:15

Operation(+, -, *, /):-

Result:74

Conclusion

Here, I was able to create a Calculator program and set up a cron job to automate the process at 1 am every day.

Bibliography

https://npu85.npu.edu/~henry/npu/classes/qa/cron_job/slide/cron.html#remove

https://npu85.npu.edu/~henry/npu/classes/shell_script/backup/slide/expect.html#calculator

Link to view the presentation

<https://docs.google.com/presentation/d/1tITnCr1U57l39P9MFyRUxngUjHwAamPilWiIjJZG7p8/edit?usp=sharing>