EMERTXE TRAINING PROJECT DOCUMENTATION FRAMEWORK

REQUIREMENTS & DESIGN DOCUMENT

Module – Linux Internals

TCP/IP Based Remote Management Solution

Contents

1 Abstract

Internet today has become a very complex entity by having different set of devices working together. In a scenario where the network devices are located remotely (ex: Wireless base station) monitoring such devices pro-actively becomes a very critical activity. Any malfunctions happen in remote device (ex: CPU usage high, Memory usage high etc..) will result in device crash. In such scenarios diagnosing problems rather than debugging is more critical. Typically such activities are done from Network Operations Center (NOC) on a 24x7x365 basis.

The goal of this project is to create a client-server based approach to get remote diagnostic information. Along with getting information, this will also need to support managing processes in the remote entity. By the end of this project developer should get a view about Linux commands, Resource management, Resource usage etc

2 Requirements

- Client and Server should communicate on TCP
- Server must be concurrent.
- User should be able to configure port number on which Server will be bound.
- User should be able to configure IP Address & Port on which Server is listening for TCP connection
- Client after connecting to Server should display a prompt say "Client] "
- Client should accept all commands at the prompt
- Following commands have to be supported by Client:

  -> get-mem <process-name> - For fetching the memory usage of given process

  -> get-cpu-usage <process-name> - For fetching CPU usage of the given process

  -> get-ports-used <process-name> - For fetching the socket ports used by given process

  -> get-open-fd <process-name> - For fetching the open file descriptors of a given process

  -> kill <process-name> - For killing a given process

  -> history - For displaying the history of commands executed for current session (Maximum of 50 commands).

  -> help (Should display available commands)

  -> "Up Arrow" / "Down Arrow" for navigating to previous commands
- Both Client and Server executable should be generated using single Makefile

3 Prerequisites

- Socket Programming

4 Design

1. Client request format (Command request)

2. Server reply format (Command reply for various commands)

3. Algorithm for handling client-only commands (History, Help etc..)
4. Keep a local array of history of commands(Max size 50). Whenever user enter a command update this array. When client exits save this contents to a file (say .remote_history). When client starts next time load this file to local array.

5 Sample Output

```
user@emertxe] ./server
RMS Server: Server Port Informantion: 127.0.0.1:6330
```

Fig 5 1: Running Server. Run till user terminates

```
user@emertxe] ./client
INFO: Connected to the Server
Type help to get supported commands
client]
```

Fig 5 2: The Client Application View

```
user@emertxe] ./client
INFO: Connected to the Server
Type help to get supported commands
client] help
get-mem <process-name>          : Gets the memory usage of given process
get-cpu-usage <process-name>    : Gets the CPU usage of given process
get-ports-used <process-name>   : Fetch the socket ports used by given process
get-open-fd <process-name>      : Fetch the open file descriptors of a given process
kill <process-name>             : Kill the given process
history                         : Show commands history
help                            : Shows this output
client]
```

Fig 5 3: help command view

```
user@emertxe] ./client
INFO: Connected to the Server
Type help to get supported commands
client] help
get-mem <process-name>          : Gets the memory usage of given process
get-cpu-usage <process-name>    : Gets the CPU usage of given process
get-ports-used <process-name>   : Fetch the socket ports used by given process
get-open-fd <process-name>      : Fetch the open file descriptors of a given process
kill <process-name>             : Kill the given process
history                         : Show commands history
help                            : Shows this output
client] get-mem firefox
INFO: Message sent to the Server
INFO: Message accepted by the Server
RESPONSE: "19.4"
client]
```

Fig 5 4: Remote Request Usage. get-mem command

```
user@emertxe] ./server
RMS Server: Server Port Informantion: 127.0.0.1:6330
RMS Server: Incoming Remote Request from 127.0.0.1:41235
```

Fig 5 5: Server Activity

```
user@emertxe] ./client
INFO: Connected to the Server
Type help to get supported commands
client] help
get-mem <process-name>          : Gets the memory usage of given process
get-cpu-usage <process-name>    : Gets the CPU usage of given process
get-ports-used <process-name>   : Fetch the socket ports used by given process
get-open-fd <process-name>      : Fetch the open file descriptors of a given process
kill <process-name>             : Kill the given process
history                         : Show commands history
help                            : Shows this output
client] get-mem firefox
INFO: Message sent to the Server
INFO: Message accepted by the Server
RESPONSE: "19.4"
client] get-ports-used firefox
INFO: Message sent to the Server
INFO: Message accepted by the Server
RESPONSE: "udp        0      0 192.168.1.102:57256     0.0.0.0:*                           47048/firefox"
client]
```

Fig 5 6: Remote Request Usage. get-ports-used command

```
user@emertxe] ./client
INFO: Connected to the Server
Type help to get supported commands
client] help
get-mem <process-name>          : Gets the memory usage of given process
get-cpu-usage <process-name>    : Gets the CPU usage of given process
get-ports-used <process-name>   : Fetch the socket ports used by given process
get-open-fd <process-name>      : Fetch the open file descriptors of a given process
kill <process-name>             : Kill the given process
history                         : Show commands history
help                            : Shows this output
client] get-mem firefox
INFO: Message sent to the Server
INFO: Message accepted by the Server
RESPONSE: "19.4"
client] get-ports-used firefox
INFO: Message sent to the Server
INFO: Message accepted by the Server
RESPONSE: "udp        0      0 192.168.1.102:57256     0.0.0.0:*                           47048/firefox"
client] history
1 get-mem firefox
2 get-ports-used firefox
client]
```

Fig 5 7: history command view

6 Artifacts

Skeleton Code

- TBD

References

- http://www.thegeekstuff.com/2012/09/SNMP-INTRODUCTION/
- http://www.cisco.com/networkers/nw04/presos/docs/NMS-1N02.pdf
- https://www.serverdensity.com/monitor/linux/how-to/
- https://www.linux.com/learn/remote-administration-linux