Project: AA 2022-2023

Computer Systems and Programming

Outline

- Development of a client-server application, which allows a client to manage files and execute commands on a remote system.
- You could develop the client either as:
  - One monolithic application, implementing all the features.
  - A set of programs, each one to perform a single function.
  - NB: syntax might be slightly different in the two cases.
- The server must be implemented as a daemon, running in background, listening on a TCP socket.

Outline

- The server allows to:
  - Copy files: **copy**
  - Move files: **move**
  - Delete files: **delete**
  - List all files/dirs (name, size, date of last access): **list**
  - Create a new directory: **create_dir**
  - Delete a directory (only if empty): **delete_dir**
  - Change the current working directory (only monolythic App.): **cd**

Example (monolithic app)

```
$ <myAppName> <remote_hostName|IP Address>
> copy file1.txt file2.txt
> delete file2.txt
> exit
$
```

Example (multiple programs)

copy *file1.txt remote_host:file2.txt*

- Copy the local file *file1.txt* to remote host *remote_host* as *file2.txt*

delete *remote_host:file2.txt*

- Remove remote file *file2.txt* on server *remote_host*

Details

- Files can be identified **both** with relative or absolute path (starting from a root directory defined in conf. file)
- Server must also be able to execute other commands, from a list contained in the configuration file, by using the command:
  - **run <*cmd*>**
  - It must be possible to create pipes, redirect output to files, etc..
  - As a default, stdin e stdout of the remote command will be those of the **run** command on the client.

Example (multiple apps)

- **run *server:cmd***
  - Runs *cmd on server* (stdout to client)
- **run *server:"cmd1 | cmd2"***
  - Runs the pipe *cmd1 | cmd2* on *server* (stdout to client)
- **run *server:"cmd > file"***
  - Runs *cmd* on *server*, output to <u>remote</u> *file*
- **run *server:cmd > file***
  - Runs *cmd* on *server*, output to <u>local</u> *file*

Example (monolithic app)

$ <myAppName> <*remote_hostName|IP Address*>

> run *cmd*

> run “*cmd1 | cmd2*”

> run “*cmd > file*”

> run *cmd > file*

> exit

$

More Details

- Actions must be performed with credentials (*uid, gid*) of the user which executed the command (**not** those of the daemon which should owned by *nobody*)
- Server must allow for multiple concurrent connections, using processes (or threads)

More Details

- Requests are received on the TCP port specified in the configuration file.
- Initial pseudo-root directory for the server is also indicated in the configuration file (already existing).
- Navigation between directories must be limited to subtree contained under the pseudo-root, not following symlinks.
- Commands contained in the configuration file must consist in only the *basename*

More Details

- Access to files must be possible maintaining consistency:
  - Single writer.
  - Multiple concurrent readers
- Command line argument for the daemon is the path to a configuration file, containing:
  - TCP/UDP port
  - Pseudo root
  - List of command basenames

Configuration File Example (YMMV)

```
#Port
45000
#Root
/user/tmp/fakeRoot
#Cmds
sort
cat
sh256sum
```

Document It!

- The project “package” must contain:
  - Source code (C, includes,..) **with comments**
  - A document file describing:
    - The design choices
    - Main modules (macrocomponents) and their implementation
    - Command syntax
    - Error Messages
    - Known Bugs (...)