Démon TCP

Table des matières

1 Introduction

Il s'agit de développer un serveur suivant des méthodes de développement éprouvée. Aussi nous utiliserons une approche modulaire permettant une construction pas à pas et ainsi la mise en place de tests unitaires de non régression.

2 Compilation croisée

Avant de tester sur la cible, on peut compiler puis tester en local, comme nous le voyons dans le préambule de ce *Makefile*.

```
targetSystem = $(shell uname -s)
targetProcessor = $(shell echo $(PATH) | grep -q 'arm-linux' && echo "arm" || echo "i386")

ifneq ($(targetSystem),Linux)
$(error unknown target system '$(targetSystem)')
endif

ifdef DEBUG
optimizingCFLAGS = -g -O0
optimizingLDFLAGS = -g
else
optimizingCFLAGS = -O2
optimizingLDFLAGS = -Xlinker -s
endif

commonCFLAGS = \
$(optimizingCFLAGS) \
-I include

CFLAG = -c
utCFLAGS = -D_utMAIN
LDFLAGS = $(optimizingLDFLAGS)
```

```
ifeq ($(targetProcessor),i386)
CC = gcc
CFLAGS = $(commonCFLAGS)
else
ifeq ($(targetProcessor),arm)
CC       =  arm-linux-gcc
C++ =  arm-linux-g++
CFLAGS   =  \
$(commonCFLAGS) \
-mcpu=arm920t \
-mtune=arm920t \
-msoft-float
else
$(error unknown target processor '$(targetProcessor)')
endif
endif
```

3 Modules

L'architecture se découpe en module. Nous les présentons dans l'ordre des dépendances. Chaque module est compilé sous forme d'objet mais également sous forme d'un exécutable permettant d'effectuer ses tests.

3.1 IO

Ce module regroupe les primitives d'entrée/sortie. Il devrait à terme présenter une abstraction des méthodes d'accès aux fichiers que ce soit via les **handler** ou les **descripteurs**. En effet, alors que l'accès aux fichiers normaux se fait via des fonctions de haut niveaux, les sockets comme d'autres fichiers spéciaux ne sont accessibles que uniquement via les fonction de bas niveau car ces accès ne sont pas buffurisés. Pour nos besoins, seule l'écriture est implémenté avec les descripteurs de fichiers.

3.2 Log

Ce module regroupe les primitives d'écriture des messages d'erreurs. Étant donné que notre Linux From Scratch ne possède pas de démon Syslog, nous limitons ici nos besoins à l'écriture dans un simple fichier de log.

3.3 Gpio

Nous avons décrit ce module dans la partie précédente.

3.4 Lexer

Bien que notre protocole soit très simple, nous utilisons le Lexer Flex pour identifier les symboles terminaux de notre grammaire.

3.5 Parser

Encore une fois, bien que notre protocole soit très simple, nous utilisons le Parser Bison pour programmer l'automate correspondant à la grammaire de notre protocole.

3.6 Serveur Tcp

Ce module écoute sur un port donné **à définir dans */etc/services***. Pour raison de commodité lors des changement d'adresses, le serveur écoute sur toutes les adresses IP. Ce module se met personnelement en relation avec les sockects clientes afin de garantir un accès exclusif au serveur.

3.7 Daemon

Ce petit bout de code à pour but de désolidariser le processus d'une session utilisateur et d'une console. Ainsi, le programme pourra tourner en tâche de fond en étant géré par les scripts d'initialisation de type System-V. Voir le fichier */etc/init.d/S99led* :

```
#!/bin/sh

start() {
        /opt/led
}

stop() {
        killall led
}

restart() {
        stop
        start
}

case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart|reload)
        restart
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac

exit $?
```

4 Tests unitaires de non régression

Il s'agit de tester chaque module indépendamment des autres ou alors dans un ordre permettant de lever les ambiguïtés sur la responsabilité de chacun des module sur les erreurs détectées. Le résultat des test est comparé au résultat attendu afin de s'assurer de ne pas avoir de régression.

```
$ make obj/io.o test/utio

$ test/utio -h
The usage for test/utio is:
```

```
        test/utio { -h | [ -i input ] [ -o output ] }   where:
                -i   : input stream
                -o   : output stream
                -h   : help

$ test/utio
text saisi recopié ci-dessous
text saisi recopié ci-dessous
^D
```