TP I – Unidad I a IV - OBLIGATORIO

Objetivo: Implementar una serie de ejercicios de programación ANSI C a modo de sintetizar toda la ejercitación propuesta en clase y en los prácticos no obligatorios de las Unidades I a IV del programa de estudios de la asignatura.

1 Implementación del programa Shell (en su versión más completa, que soporta procesos foreground y background; que haga uso de la señal SIGCHLD para verificar la finalización de procesos en background) que evite la existencia de procesos zombies. El programa termina cuando el usuario ingresa el comando "salir". Si el usuario presiona CTRL+C el programa esperará a que terminen todos los procesos y luego finalizará sin dejar procesos zombies o huérfanos.

Links de ayuda para este ejercicio:
Minishell 01
https://www.youtube.com/watch?v=8eZUwHpPVjE

C Linux - Relación entre wait y exit - 05
https://www.youtube.com/watch?v=KKmqjPQu3Oo

C Linux Señales 01
https://www.youtube.com/watch?v=RqGh52B5B5A

C Linux Señales 02 Ejemplo
https://www.youtube.com/watch?v=3mkP_GAxKfs

Klingon Vs. La Federación Interestelar (Señales y Conjunto de Señales)
https://www.youtube.com/watch?v=ZzXqR1VVBOU

2 Implementar un proceso al cual se le indique por línea de comando la cantidad de procesos a crear, todos los procesos a crear serán hermanos; cada uno de ellos retornará un valor entero distinto al proceso padre. Los procesos hijos quedan en un loop eterno, en una espera no activa, cuando recibe la señal SIGUSR1 o SIGINT el proceso hijo termina retornando un valor entero distinto al de sus hermanos. El proceso padre reportará por pantalla la sumatoria de los retornos de los procesos hijos creados. No se permitirá que existan procesos huérfanos o zombies.

3 Implementación de una sincronización de los hilos A, B y C de forma tal, que la secuencia de ejecución y acceso a su sección crítica sea la siguiente: ABAC... detener el proceso luego de N iteraciones completas (el número N se ingresa por línea de comandos). Resolver la sincronización con variables Mutex (librería pthread).

4 Implementación de una sincronización con procesos independientes A, B y C de forma tal, que la secuencia de ejecución y acceso a su sección crítica sea la siguiente: ABAC... detener el proceso luego de N iteraciones completas (el número N se ingresa por línea de comandos). Resolver la sincronización con: semáforos Posix con nombre.

5 Idem anterior resolviendo la sincronización con semáforos Posix sin nombre.

6 Idem anterior resolviendo la sincronización con semáforos SVR4.

Links de ayuda para ejercicios 4, 5, 6:

Sincronización BACA
https://www.youtube.com/watch?v=UZLaeemleRk

C Linux Semáforos System V 01
https://www.youtube.com/watch?v=ub2YTpCq3Aw

C Linux Semáforos System V Ejemplo 02
https://www.youtube.com/watch?v=vvI2-CZ1Bik

C Linux Semáforos System V Otros Dos Ejemplos 03
https://www.youtube.com/watch?v=Gy4baWme_ek

C Linux Semáforos System V ABoCDoE 07
https://www.youtube.com/watch?v=vLZt2yYjX6w

Semáforos POSIX Sin Nombre en Procesos Independientes, ejemplo en lenguaje C
https://www.youtube.com/watch?v=T6ZOOxhDSe8

7 Implementación de una sincronización con procesos emparentados PadreA, HijoB y HijoC de forma tal, que la secuencia de ejecución y acceso a su sección crítica sea la siguiente: PadreAHijoBPadreAHijoC... detener el proceso luego de N iteraciones completas (el número N se ingresa por línea de comandos). HijoB e HijoC son hermanos. Resolver la sincronización de la forma que a Ud. le parezca más apropiada (no usar señales).

8 Dada una matriz de números enteros M de 4 filas y 3 columnas, dado el vector T de 3 elementos que guardará la sumatoria de cada columna para todas las filas, cada elemento del vector T esta inicializado con -1.
El programa principal lanza los siguientes hilos:
hiloQueLlenaMatriz -> hilo que utiliza función random() y llena la matriz con números enteros aleatorios de dos cifras
hiloQueSumaColumna1 -> no puede comenzar su ejecución hasta que hiloQueLlenaMatriz finalice, guarda total en T[0] y chequea los otros totales de T, si están todos disponibles, entonces señala variable de condición V
hiloQueSumaColumna2 -> no puede comenzar su ejecución hasta que hiloQueLlenaMatriz finalice, , guarda total en T[1] y chequea los otros totales de T, si están todos disponibles, entonces señala variable de condición V
hiloQueSumaColumna3 -> no puede comenzar su ejecución hasta que hiloQueLlenaMatriz finalice, , guarda total en T[2] y chequea los otros totales de T, si están todos disponibles, entonces señala variable de condición V
hiloQueMuestraTotalGeneral -> muestra la sumatoria de T[0..2] , este hilo espera por la variable de condición V

9 Realizar un programa que reciba por línea de comandos un comando a ejecutar y sus argumentos. El programa va crear un proceso hijo con dicho comando y el proceso padre leerá la salida del comando. Para lograr esto, cree un pipe entre proceso padre e hijo y utilice la función dup() o dup2() para duplicar la salida en el pipe y luego leer la salida usando el pipe. No está permitido usar la función popen() que resume o simplifica la técnica propuesta. No generar procesos

huérfanos ni zombies.

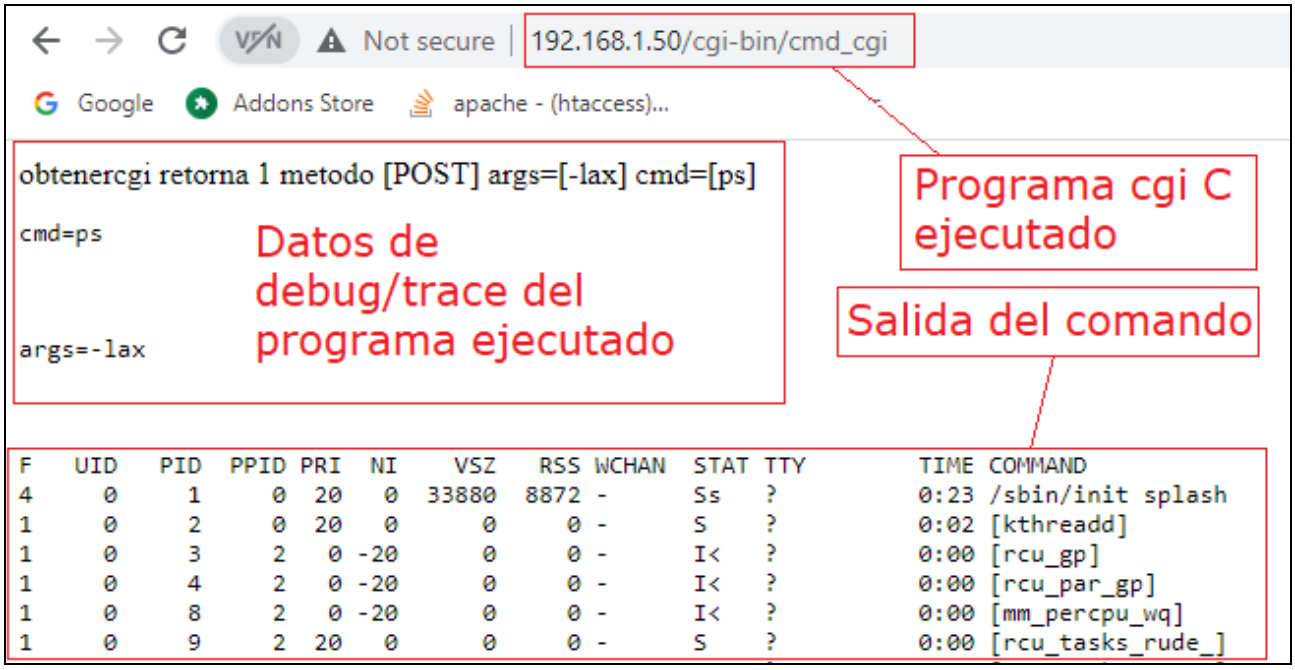
Links de ayuda para este ejercicio:
Pipes 1
https://www.youtube.com/watch?v=WigeRo_m6VE

Pipes 2
https://www.youtube.com/watch?v=XM8DbmgI5TE

Pipes 3
https://www.youtube.com/watch?v=tXJMeasmlI8

10 Implementar un Sistema de Monitoreo Distribuido. A través de una serie de comandos (ls, ipcs, df, ps, free, etc) podemos monitorear la actividad que está llevando adelante el sistema operativo. Se propone hacer una aplicación web muy sencilla, en cuya página index.html permita tener varios links para ejecutar distintos comandos del sistema operativo que aporten información al operador del sistema, por ejemplo, supongamos que el computador a monitorear tiene la IP 192.168.1.50:

Entonces, cuando el operador ingresa en la opción Procesos, se podría ejecutar un programa CGI "genérico" que sirva para cualquier comando Linux (recibe –ya sea por método POST o GET- dos parámetros: "cmd" –que indica el comando a ejecutar- y "args" –que indica los argumentos del comando a ejecutar-) o variantes con distintos argumentos de un mismo comando. Aquí mostramos una salida posible, indicando algunos datos adicionales que usamos en la prueba del programa CGI:



De esta forma, podemos implementar muchísimos comandos de monitoreo para el sistema operativo y si tuviésemos muchos computadores por controlar podríamos hacer otra página web que agrupe las distintas IP's a controlar, entonces, una empresa podría monitorear varios servidores Linux/Unix con una única aplicación distribuida:

Realmente se pueden hacer muchas variantes de esta idea, incluso podríamos monitorear sistemas Windows (por ejemplo: instalar cygwin o mingw + servidor XAMPP (solo usaríamos Apache web server, también podría usarse MS-IIS express, etc.) + programa CGI compilado con cygwin o mingw).
Una implementación posible para Linux:
-Instalar servidor web lighttpd https://www.lighttpd.net

```
$ sudo apt-get install lighttpd
```

-Configurar lighttpd para ejecutar programas CGI:

-copiar configuración CGI:

```
$ sudo cp /etc/lighttpd/conf-available/10-cgi.conf
/etc/lighttpd/conf-enabled
```

-editar o cambiar archivo de configuración CGI si fuese necesario (mostramos contenido):

```
pi@raspberrypi: ~/2022/tp
pi@raspberrypi:~/2022/tp $ cat /etc/lighttpd/conf-enabled/10-cgi.conf
# /usr/share/doc/lighttpd/cgi.txt

server.modules += ( "mod_cgi" )

$HTTP["url"] =~ "^/cgi-bin/" {
        cgi.assign = ( "." => "/bin/sh",".sh" => "/bin/sh","" => "" )
        alias.url += ( "/cgi-bin/" => "/usr/lib/cgi-bin/" )
}

## Warning this represents a security risk, as it allow to execute any file
## with a .pl/.py even outside of /usr/lib/cgi-bin.
#
#cgi.assign      = (
#       ".pl"  => "/usr/bin/perl",
#       ".py"  => "/usr/bin/python",
#)
pi@raspberrypi:~/2022/tp $
```

-editar o cambiar archivo de configuración de lighttpd (mostramos contenido):

pi@raspberrypi: ~/2022/tp

```
pi@raspberrypi:~/2022/tp $ cat /etc/lighttpd/lighttpd.conf
server.modules = (
        "mod_indexfile",
        "mod_access",
        "mod_alias",
        "mod_redirect",
        "mod_cgi",
)

server.document-root        = "/var/www/html"
server.upload-dirs          = ( "/var/cache/lighttpd/uploads" )
server.errorlog             = "/var/log/lighttpd/error.log"
server.pid-file             = "/run/lighttpd.pid"
server.username             = "www-data"
server.groupname            = "www-data"
server.port                 = 80

# features
#https://redmine.lighttpd.net/projects/lighttpd/wiki/Server_feature-flagsDetails
server.feature-flags       += ("server.h2proto" => "enable")
server.feature-flags       += ("server.h2c"     => "enable")
server.feature-flags       += ("server.graceful-shutdown-timeout" => 5)
#server.feature-flags       += ("server.graceful-restart-bg" => "enable")

# strict parsing and normalization of URL for consistency and security
# https://redmine.lighttpd.net/projects/lighttpd/wiki/Server_http-parseoptsDetai
ls
#  (might need to explicitly set "url-path-2f-decode" = "disable"
#   if a specific application is encoding URLs inside url-path)
server.http-parseopts = (
  "header-strict"           => "enable",# default
  "host-strict"             => "enable",# default
  "host-normalize"          => "enable",# default
  "url-normalize-unreserved"=> "enable",# recommended highly
  "url-normalize-required"  => "enable",# recommended
  "url-ctrls-reject"        => "enable",# recommended
  "url-path-2f-decode"      => "enable",# recommended highly (unless breaks app)
 #"url-path-2f-reject"      => "enable",
  "url-path-dotseg-remove"  => "enable",# recommended highly (unless breaks app)
 #"url-path-dotseg-reject"  => "enable",
 #"url-query-20-plus"       => "enable",# consistency in query string
)

index-file.names            = ( "index.php", "index.html" )
url.access-deny             = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )

# default listening port for IPv6 falls back to the IPv4 port
include_shell "/usr/share/lighttpd/use-ipv6.pl " + server.port
include_shell "/usr/share/lighttpd/create-mime.conf.pl"
include "/etc/lighttpd/conf-enabled/*.conf"

#server.compat-module-load   = "disable"
server.modules += (
        "mod_dirlisting",
        "mod_staticfile",
)
pi@raspberrypi:~/2022/tp $
```

Activar el modulo CGI

carpeta de paginas web

Parametros principales

-Probar lighttpd:

-copiar página web index.html en /var/www/html (use sudo o usuario root)

-usar tag a o formulario para ejecutar programa CGI pasando los parámetros apropiados (acorde con el comando Linux que quiera ejecutar)

-copiar programa CGI ejecutable a /usr/lib/cgi-bin (use sudo o usuario root)

-abrir navegador web (en cualquier pc de la red) y escribir la ip del servidor Linux. Si todo ok, debería ver página index.html en su navegador web y de allí poder ejecutar su programa CGI

Básicamente se trata de instalar un web server en cada computador a monitorear + página index.html + programa CGI C que permita ejecutar comandos, leer la salida de los comandos (como se hizo en el ejercicio anterior) y mostrarlo en página web.

Si bien existen hoy día frameworks de desarrollo web muy sofisticados y más productivos que este, con la interfaz CGI se podría implementar cualquier aplicación web[1].
En clase vamos a orientar y ayudar para la implementación de este ejercicio, también proponemos una discusión en torno a la interfaz CGI.

[1] Incluso, utilizando librerías Javascript, tales como jQuery, se puede implementar una aplicación de tipo SPA (single page application), con interfaz gráfica dinámica y compleja, en donde no es necesario recargar la página desde el servidor para lograr una interacción fluida con el usuario.

ANEXO

Configuración CGI y FastCGI de servidor web lighttpd (probado en Linux ARM Raspberry pi 4):

Instalar servidor lighttpd:

```
$ sudo apt-get install lighttpd
```

Arrancar servidor

```
$ sudo /etc/init.d/lighttpd start
```

También se puede hacer:

```
$ sudo systemctl start lighttpd.service
```

Detener servidor

```
$ sudo /etc/init.d/lighttpd stop
```

También se puede hacer:

```
$ sudo systemctl stop lighttpd.service
```

Re-arrancar servidor

```
$ sudo /etc/init.d/lighttpd restart
```

También se puede hacer:

```
$ sudo systemctl restart lighttpd.service
```

Cada vez que modifique un archivo de configuración, deberá reiniciar el servidor para que tome los cambios.

Archivo **/etc/lighttpd/lighttpd.conf**

```
server.modules = (
      "mod_indexfile",
      "mod_access",
      "mod_alias",
      "mod_redirect",
      "mod_cgi",
      "mod_fastcgi",
)

server.document-root        = "/var/www/html"
server.upload-dirs          = ( "/var/cache/lighttpd/uploads" )
server.errorlog             = "/var/log/lighttpd/error.log"
server.pid-file             = "/run/lighttpd.pid"
server.username             = "www-data"
server.groupname            = "www-data"
server.port                 = 80

# features
#https://redmine.lighttpd.net/projects/lighttpd/wiki/Server_feature-flagsDetails
server.feature-flags       += ("server.h2proto" => "enable")
server.feature-flags       += ("server.h2c"     => "enable")
server.feature-flags       += ("server.graceful-shutdown-timeout" => 5)
#server.feature-flags       += ("server.graceful-restart-bg" => "enable")

# strict parsing and normalization of URL for consistency and security
# https://redmine.lighttpd.net/projects/lighttpd/wiki/Server_http-
parseoptsDetails
```

```
# (might need to explicitly set "url-path-2f-decode" = "disable"
#  if a specific application is encoding URLs inside url-path)
server.http-parseopts = (
  "header-strict"           => "enable",# default
  "host-strict"             => "enable",# default
  "host-normalize"          => "enable",# default
  "url-normalize-unreserved"=> "enable",# recommended highly
  "url-normalize-required"  => "enable",# recommended
  "url-ctrls-reject"        => "enable",# recommended
  "url-path-2f-decode"      => "enable",# recommended highly (unless breaks app)
 #"url-path-2f-reject"      => "enable",
  "url-path-dotseg-remove"  => "enable",# recommended highly (unless breaks app)
 #"url-path-dotseg-reject"  => "enable",
 #"url-query-20-plus"       => "enable",# consistency in query string
)

index-file.names            = ( "index.php", "index.html" )
url.access-deny             = ( "~", ".inc" )
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )

# default listening port for IPv6 falls back to the IPv4 port
include_shell "/usr/share/lighttpd/use-ipv6.pl " + server.port
include_shell "/usr/share/lighttpd/create-mime.conf.pl"
include "/etc/lighttpd/conf-enabled/*.conf"

#server.compat-module-load   = "disable"
server.modules += (
      "mod_dirlisting",
      "mod_staticfile",
)
```

Aquí se puede observar que este servidor será ejecutado por el usuario www-data que pertenece al grupo www-data . Si Ud pretende ejecutar programas CGI, FastCGI, etc. debe tener en cuenta que las carpetas y los programas deben ser ejecutables (tener los permisos suficientes) para este usuario.

Archivo **/etc/lighttpd/conf-enabled/10-cgi.conf**

```
# /usr/share/doc/lighttpd/cgi.txt

server.modules += ( "mod_cgi" )

$HTTP["url"] =~ "^/cgi-bin/" {
      cgi.assign = ( "." => "/bin/sh",".sh" => "/bin/sh","" => "" )
      alias.url += ( "/cgi-bin/" => "/usr/lib/cgi-bin/" )
}

## Warning this represents a security risk, as it allow to execute any file
## with a .pl/.py even outside of /usr/lib/cgi-bin.
#
#cgi.assign      = (
#     ".pl"  => "/usr/bin/perl",
#     ".py"  => "/usr/bin/python",
#)
```

En nuestro caso es relevante que toda URL que contenga a /cgi-bin/ terminará invocando un programa CGI que se encuentre en la carpeta /usr/lib/cgi-bin/

Archivo **/etc/lighttpd/conf-enabled/15-fastcgi.conf**

```
# /usr/share/doc/lighttpd/fastcgi.txt.gz
#
http://redmine.lighttpd.net/projects/lighttpd/wiki/Docs:ConfigurationOptions#mod
_fastcgi-fastcgi

fastcgi.debug = 1
server.modules += ( "mod_fastcgi" )
fastcgi.server = (
  "/test.cgi" =>
  ( "test.cgi" => (
      "host" => "127.0.0.1",
      "port" => 8000,
      "check-local" => "disable",
      "docroot" => "/usr/lib/cgi-bin",
      "bin-path" => "/usr/lib/cgi-bin/test.cgi",
      "max-procs" => 1
  ))
)
```

El archivo 15-fastcgi.conf configura FastCGI en servidor lighttpd. Cuando el usuario tipea una URL en su navegador web que incluye a /test.cgi (ejemplo: supongamos que nuestro servidor lighttpd se encuentra en la IP 192.168.1.50, la URL sería http://192.168.1.50/test.cgi) entonces lighttpd tomará la petición del usuario y la re-enviará a la dirección IP 127.0.0.1 (dirección de "localhost", pero bien podría ser otra IP dentro de esta red, permitiendo distribuir la carga de trabajo del servidor), puerto 8000 (también se podría haber usado cualquier otro puerto TCP/IP disponible, suele usarse también el puerto 9000), en esa dirección, quien responderá a dicha petición será el programa FastCGI /usr/lib/cgi-bin/test.cgi . El servidor lighttpd –con esta configuración- hará automáticamente un spawn (fork() + exec()) del programa /usr/lib/cgi-bin/test.cgi[2] para hacer esta tarea utiliza un programa que viene con el servidor lighttpd que se llama spawn-fcgi (/usr/bin/spawn-fcgi). No es relevante el nombre "test.cgi" sirve solo a los efectos del log de errores del servidor lighttpd en donde se incluirá en los mensajes dicho nombre, permitiendo nuestra orientación.
Si deseamos incluir otro camino o path a ser tratado por otro programa FastCGI, debemos agregar otra entrada con dicho path en este archivo de configuración de la siguiente forma:

```
fastcgi.server = (
  "/test.cgi" =>
  ( "test.cgi" => (
      "host" => "127.0.0.1",
      "port" => 8000,
      "check-local" => "disable",
      "docroot" => "/usr/lib/cgi-bin",
      "bin-path" => "/usr/lib/cgi-bin/test.cgi",
      "max-procs" => 1
  )),
  "/otro.cgi" =>
  ( "otro.cgi" => (
      "host" => "127.0.0.1",
```

[2] Con esta configuración, cuando Ud arranca el servidor lighttpd, puede hacer un $ ps ax y ver el proceso del servidor lighttpd, cuando se invoque una URL de un programa FastCGI ejecute nuevamente $ ps ax y podrá ver el spawn del programa FastCGI, si continua haciendo invocaciones de URL's del mismo programa FastCGI podrá observar que no se crean nuevos procesos, siempre será el mismo proceso –previamente cargado en memoria- quien atenderá las peticiones, evitando hacer un fork() por cada petición como sucede con los programas CGI. Si Ud quiere detener el programa FastCGI spawned envíe al mismo señal SIGUSR1 y podrá observar como lighttpd vuelve a hacer otro spawn del programa FastCGI.

```
      "port" => 8001,
      "check-local" => "disable",
      "docroot" => "/usr/lib/cgi-bin",
      "bin-path" => "/usr/lib/cgi-bin/otro.cgi",
      "max-procs" => 1
  ))
)
```

En este caso, cuando el usuario tipea http://192.168.1.50/test.cgi será tratado por el programa FastCGI test.cgi y cuando se tipea http://192.168.1.50/otro.cgi será tratado por el programa FastCGI otro.cgi . Obsérvse que se utilizó un puerto TCP/IP distinto, pues no puedo tener más de un proceso escuchando en el mismo puerto, en el mismo host.
Hay diferencias entre un programa CGI y otro programa FastCGI, pertenecen a interfaces diferentes, tema para discutir en clase.

Contenido de carpeta **/etc/lighttpd**

```
total 12
drwxr-xr-x 2 root root 4096 mar  1 18:52 conf-available
drwxr-xr-x 2 root root 4096 jul  6 16:20 conf-enabled
-rw-r--r-- 1 root root 2249 jul  5 01:46 lighttpd.conf
```

Puede observarse que hay un único archivo de configuración lighttpd.conf con los parámetros generales del servidor. Luego hay una carpeta conf-available que contiene todos los archivos de configuración disponibles para cada uno de los módulos del servidor a configurar:

Contenido de carpeta **/etc/lighttpd/conf-available**

```
total 116
-rw-r--r-- 1 root root 919 ene 10 04:38 05-auth.conf
-rw-r--r-- 1 root root 725 ene 10 04:38 05-setenv.conf
-rw-r--r-- 1 root root  91 feb 17  2021 10-accesslog.conf
-rw-r--r-- 1 root root 396 feb 17  2021 10-cgi.conf
-rw-r--r-- 1 root root  63 feb 17  2021 10-dir-listing.conf
-rw-r--r-- 1 root root  36 feb 17  2021 10-evasive.conf
-rw-r--r-- 1 root root 128 feb 17  2021 10-evhost.conf
-rw-r--r-- 1 root root 104 feb 17  2021 10-expire.conf
-rw-r--r-- 1 root root 177 feb 17  2021 10-fastcgi.conf
-rw-r--r-- 1 root root  42 feb 17  2021 10-flv-streaming.conf
-rw-r--r-- 1 root root  82 feb 17  2021 10-no-www.conf
-rw-r--r-- 1 root root 849 feb 17  2021 10-proxy.conf
-rw-r--r-- 1 root root 176 feb 17  2021 10-rewrite.conf
-rw-r--r-- 1 root root 253 feb 17  2021 10-rrdtool.conf
-rw-r--r-- 1 root root 398 feb 17  2021 10-simple-vhost.conf
-rw-r--r-- 1 root root 449 feb 17  2021 10-sockproxy.conf
-rw-r--r-- 1 root root  99 feb 17  2021 10-ssi.conf
-rw-r--r-- 1 root root 353 ene 10 04:38 10-ssl.conf
-rw-r--r-- 1 root root 460 feb 17  2021 10-status.conf
-rw-r--r-- 1 root root 450 feb 17  2021 10-userdir.conf
-rw-r--r-- 1 root root  38 feb 17  2021 10-usertrack.conf
-rw-r--r-- 1 root root 168 ene 10 04:37 11-extforward.conf
-rw-r--r-- 1 root root 570 ene 10 04:38 15-fastcgi-php.conf
-rw-r--r-- 1 root root 355 ene 10 04:38 15-fastcgi-php-fpm.conf
-rw-r--r-- 1 root root 248 ene 10 04:38 20-deflate.conf
-rw-r--r-- 1 root root 508 feb 17  2021 90-debian-doc.conf
-rw-r--r-- 1 root root  56 jul 28  2013 90-javascript-alias.conf
-rw-r--r-- 1 root root 162 feb 17  2021 99-unconfigured.conf
-rw-r--r-- 1 root root 843 feb 17  2021 README
```

Estos archivos son archivos de configuración disponibles para cada módulo a configurar, si queremos activar un módulo, debemos modificar /etc/lighttpd/lighttpd.conf para que dicho modulo

sea cargado y debemos copiar el archivo de configuración del módulo de la carpeta /etc/lighttpd/conf-available a la carpeta /etc/lighttpd/conf-enabled y modificar el archivo acorde a nuestras necesidades dentro de la carpeta /etc/lighttpd/conf-enabled

Contenido de carpeta **/etc/lighttpd/conf-enabled**

```
total 8
-rw-r--r-- 1 root root 432 mar  1 19:14 10-cgi.conf
-rw-r--r-- 1 root root 493 jul  6 16:20 15-fastcgi.conf
lrwxrwxrwx 1 root root  42 ene 11  2021 90-javascript-alias.conf -> ../conf-
available/90-javascript-alias.conf
```

Los archivos resaltados fueron copiados de conf-available a conf-enabled y luego modificados, el archivo no resaltado no fue modificado y fue puesto por el programa instalador del servidor.

Contenido de la carpeta **/var/log/lighttpd**

```
total 124
-rw-r--r-- 1 www-data www-data 71021 jul  6 20:57 error.log
-rw-r--r-- 1 www-data www-data   969 jun 28 22:44 error.log.1
-rw-r--r-- 1 www-data www-data   169 abr 15 03:52 error.log.10.gz
-rw-r--r-- 1 www-data www-data    87 abr  3 00:01 error.log.11.gz
-rw-r--r-- 1 www-data www-data    87 abr  1 15:02 error.log.12.gz
-rw-r--r-- 1 www-data www-data   196 jun 11 10:04 error.log.2.gz
-rw-r--r-- 1 www-data www-data    88 may 29 00:01 error.log.3.gz
-rw-r--r-- 1 www-data www-data    88 may 22 00:01 error.log.4.gz
-rw-r--r-- 1 www-data www-data    87 may 15 00:01 error.log.5.gz
-rw-r--r-- 1 www-data www-data    88 may  8 00:01 error.log.6.gz
-rw-r--r-- 1 www-data www-data    85 may  1 00:00 error.log.7.gz
-rw-r--r-- 1 www-data www-data   133 abr 27 14:17 error.log.8.gz
-rw-r--r-- 1 www-data www-data   190 abr 23 17:15 error.log.9.gz
```

El archivo error.log contiene todos los mensajes que emite el servidor lighttpd, es una fuente de información muy importante cuando tenemos algún problema de configuración o de ejecución. Es muy importante implementar algún archivo log dentro de los programas CGI o FastCGI para grabar allí mensajes que nos orienten en cuanto a la traza del programa y poder comprobar los valores de las variables y mensajes de error.

Instalar librería fcgi para la creación de programas FastCGI:

```
$ sudo apt-get install libfcgi libfcgi-dev
```

Como compilar programas FastCGI en C:

```
$ gcc -Wall -o test.cgi test_cgi.c -lfcgi
```

Ejemplo de programa FastCGI en C (test_cgi.c):

```
#include <fcgi_stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv) {
      char host[200];
      char *p = getenv("SERVER_NAME");
      if ( p != NULL )  strcpy(host,p);
      else              strcpy(host,"DESCONOCIDO");
      // loop FastCGI
      while (FCGI_Accept() >= 0) {
            printf("Status: 200 OK\r\n");
            printf("Content-type: text/html\r\n\r\n");
            printf("<!doctype html><html><body>\n");
            printf("<p>Hola mundo from fastcgi with lighttpd!</p>\n");
            printf("<p>Host: [%s]</p>\n",host);
            printf("</body></html>\n");
      }
      return 0;
}
```

Hay varias consideraciones para hacer en cuanto a código FastCGI, no es lo mismo que CGI, obsérvese que la salida del programa comienza con “Status..” esto no es así ni es requerido por los programas CGI. La salida de un programa CGI es lo que se retorna al cliente, sin embargo, la salida de un programa FastCGI es tomada y modificada por el servidor web antes de ser devuelto al cliente. No incluir valor NULL en la salida. No “resumir printf’s” por ejemplo, printf(“Status ..\r\nContent-type:…\r\n”); no hacerlo; en vez de ello, hacer dos printf’s uno para imprimir “Status..\r\n” y otro para “Content-type..\r\n” en forma individual. Todas las tareas a realizarse una única vez, deben hacerse antes del loop principal del programa.

Instalar programa ejecutable FastCGI para ser ejecutado por servidor lighttpd:

```
$ sudo cp test.cgi /usr/lib/cgi-bin
```

Debe asegurarse que el usuario www-data tiene privilegios en la carpeta /usr/lib/cgi-bin , que, al menos, puede ejecutar estos programas:

Contenido de la carpeta **/usr/lib/cgi-bin**

```
total 448
-rwxr-xr-x 1 root root  22236 jun 28 23:14 abmprod
-rwxr-xr-x 1 root root   8488 mar 16 23:45 checklock
-rwxr-xr-x 1 root root  12948 mar 13 21:35 cmd_cgi
-rwxr-xr-x 1 root root  13140 mar 13 21:35 cmd_cgi2
-rwxr-xr-x 1 root root   8260 abr 23 20:22 ipcborrar
-rwxr-xr-x 1 root root  13644 abr 23 20:22 ipccrear
-rwxr-xr-x 1 root root  13808 jun 28 23:11 lockprod
-rwxr-xr-x 1 root root  13720 may  5 19:26 login
-rwxrwxrwx 1 root root  28599 jun 27 17:52 login.txt
-rwxr-xr-x 1 root root  13668 abr 23 19:04 logoff
-rwxr-xr-x 1 root root  12912 jun  5 20:45 menucgi
-rwxrwxrwx 1 root root 224000 jun 28 23:16 prod.dat
-rwxrwxrwx 1 root root   1785 abr  4 16:13 prod.txt
-rwxr-xr-x 1 root root  13748 jun 28 22:16 selprod
-rwxr-xr-x 1 root root   8444 jul  6 16:38 test.cgi
-rwxrwxrwx 1 root root     73 abr 23 16:38 usuario.txt
```

Obsérvese que el dueño de estos archivos es root (y no el usuario www-data), sin embargo, tanto root, como el grupo al que pertenece root, como el resto de los usuarios, tienen permiso de ejecución (x). Obsérvese que también hay archivos de datos (no es recomendable usar esta carpeta

para datos) y como se debe leer y grabar en los mismos, el resto de los usuarios tiene privilegios de lectura y escritura sobre dichos archivos, de esta forma www-data podrá ejecutar los programas CGI y FastCGI y además leer y grabar en los archivos de datos.