Gopher Relazione - Programmazione Di Sistema 2018/2019

Andrea Bacciu & Valerio Neri

Indice

Capitolo 1

Introduzione

1.1 File Di Configurazione E Paramatri Da Linea Di Comando

Il file di configurazione è un file di testo (.txt) denominato 'gopher_server_configuration.txt' il quale deve riportare esattamente 4 righe con i relativi campi, non duplicati, {mode_concurrency, port, root_dir, external_ip} l'ordine di scrittura dei campi è indifferente.

1. **root_dir** - Sono richieste delle doppie virgolette per il path, nel quale possono essere presenti spazi. I percorsi sono dipendenti dal sistema operativo. Per utilizzare il parametro root_dir da riga di comando è richiesto l'uso dell'opzione -d seguita dal valore. È richiesta una root_dir di dimensioni inferiori a 2048 byte.

2. **mode_concurrency** - Può assumere i due valori { Thread, Process }. Per utilizzare il parametro mode_concurrency da riga di comando è richiesto l'uso dell'opzione -m seguita dal valore.

3. **port** - Può assumere i valori da 0 a 65535, di default la porta scelta è la 7070 che non richiede permessi di amministratore per essere utilizzata. Per utilizzare il parametro port da riga di comando è richiesto l'uso dell'opzione -p seguita dal valore.

4. **external_ip** - È richiesto un hostname di dimensioni inferiori a 2048 byte comprensivo dei doppi apici. Per utilizzare il parametro external_ip da riga di comando è richiesto l'uso dell'opzione -i seguita dal valore.

5. **DOS_protection** - Opzione da riga di comando attivabile con -s (security), non sono richiesti argomenti. L'opzione setta a *True* la variabile *DOS_PROTECTION*.

6. **Modalità Deamon** La modalità Deamon è una funzione dedicata alla versione Linux. La modalità Deamon è attivabile tramite l'opzione -e, senza ulteriori argomenti, da linea di comando.

In caso di errori nel file di configurazione, il programma terminerà immediatamente ritornando un errore.

1.2 Compilazione

Il compilatore da noi scelto per la piattaforma Windows è Minimalist GNU for Windows (Mingw), mentre per la piattaforma Linux e Mac abbiamo utilizzato GNU Compiler Collection (GCC).

La compilazione del progetto é supportata da CMake, ed é unica sia per Windows che per Linux. Su Linux viene generato il file binario, mentre su Windows verranno generati tre eseguibili.

```
cmake
make
```

Purché la compilazione abbia successo, è necessario che su Linux siano disponibili le seguenti librerie:

1. **libpthread** - POSIX threads library.
2. **librt** - POSIX.1b Realtime Extensions library.

1.3 Struttura Delle Cartelle

Il progetto é diviso in tre parti, ognuna contenuta nell'omonima cartella nella root del progetto:

1. **general** - Contenente il codice condiviso sia da Linux che da Windows.
2. **linux** - Contenente il codice specifico per la piattaforma Linux.
3. **windows** - Contenente il codice specifico per la piattaforma Windows.

Ognuna di queste cartelle contiene una cartella **include** con tutti gli header file ed una cartella **lib**, la quale é a sua volta divisa in diverse cartelle, ognuna delle quali contiene il codice sorgente per una determinata area del progetto.

Nella root del progetto sono presenti:

1. **main.c** - il main del progetto
2. **main2.c** - il main per il processo che su Windows gestirà il file di log.
3. **main_HandleRequest.c** - il main per il processo che su Windows gestirà la richiesta ricevuta dal client.
4. **gopher_log_file.txt** - il file di log
5. **gopher_server_configuration.txt** - il file di configurazione
6. **stressCurl.sh** - file bash per il testare il server tramite CURL

1.4 Definizioni

Il server utilizza diverse costanti numerici e valore di convenzione, sono tutte definite nel file **definitions.h**. Esempi di definizioni contenute in questo file sono i percorsi ai file di configurazione e di log, la dimensione dei buffer, i separatori per i diversi sistemi operativi e costanti numeriche per indicare il tipo di file e costanti numeriche per indicare eventuali valori di ritorno.

1.5 Codifica Del Tipo Di File

Quando al server viene richiesto un file assegnerà il codice gopher in base all'output del comando *file* per Linux ed in base all'estensione per Windows. Sono stati inseriti nel server valori per riconosce alcuni tipi di file.

Capitolo 2

General

2.1 Detached

É stato scelto di non controllare il termine dell'esecuzione di thread e processi, cosi facendo il thread principale rimane sempre in attesa per nuove richieste di connessione, evitando che si fermi ad attendere la terminazione di thread o processi rischiando cosi di perdere richieste di connessione.

É stato scelto di non fare un pool di richieste dato che non é possibile assumere il tempo necessario per il soddisfacimento di una richiesta.

2.2 Gestione Dell'Aggiornamento Del File Di Configurazione

Può essere richiesto l'aggiornamento del file di configurazione tramite un segnale *SIGHUP* per Linux, ed un evento console *CTRL-C* per Windows. Quando il server gestirà il segnale o l'evento verrà impostato un flag per indicare la volontà dell'utente di aggiornare la configurazione.

Il controllo del flag e l'aggiornamento effettivo della configurazione del server avviene in modo diverso nei due Sistemi Operativi, come descritto di seguito.

Linux - Per aggiornare la configurazione viene sfruttata la capacità della select di bloccarsi alla ricezione del segnale e settando la variabile errno uguale alla costante EINTR.

Windows - Il server si accorgerà della ricezione dell'evento al termine di un timeout impostato sulla select, il flag é stato settato allora si procederà all'aggiornamento.

L'aggiornamento effettivo della configurazione del server avviene chiudendo il file descriptor della socket del server e riaprendolo con la nuova configurazione, lasciando continuare l'esecuzione a thread o processi che stanno risolvendo richieste precedenti.

2.3 Log

2.3.1 File Log

Il file log un file di testo (.txt) si trova nella directory root del progetto denominato 'gopher_log_file.txt'. All'interno sono registrate IP del client, dimensioni e percorso di ogni file richiesto.

2.3.2 Processo Log

Windows

Il Thread o Processo che gestisce la richiesta crea un nuovo processo Log in attesa di un evento. Quando il processo viene risvegliato, legge dalla Pipe, scrive una riga di log nel file che gli é stata passata dal processo che gestisce la richiesta, termina.

Linux

All'avvio del server viene creato un processo sempre attivo in attesa su una condition variable. Quando viene risvegliato il processo legge dalla Pipe, scrive una riga di log nel file che gli é stata passata dal processo che gestisce la richiesta, torna in attesa sulla condition variable.

Dato che la *signal* della condition variable non ha effetto se non é presente nessun processo in attesa su essa, é stato necessario sincronizzare i due processi, abbiamo scelto di sincronizzarli tramite una pipe. In questa situazione, il processo che vuole risvegliare la condition variable legge su una pipe, sulla quale scriverà il processo che attenderà sulla condition variable dopo aver preso il mutex, cosi da evitare problemi di concorrenza.

2.3.3 Condition Variable Con Memory Mapping

Su Linux prima della *fork* del processo log verrà creata una shared memory tra i due processi in modo da condividere il mutex e la condition variable necessari per la sincronizzazione.

Per garantire la sezione critica abbiamo utilizzato i mutex e le condition variable della libreria pthread, specifiche a 1.2. Sia i mutex che le condition variable consentono, se correttamente inizializzate con l'attributo *PTHREAD_PROCESS_SHARED*, di essere utilizzate fra thread, anche di diversi processi. All'interno della sezione critica i processi comunicheranno le informazioni da registrare nel file di log tramite una pipe semplice.

Capitolo 3

Sistemi Operativi Non Supportati E Note Aggiuntive

3.1 Supporto MacOS

Il server non é compatibile con il sistema operativo di Apple dato che, sebbene sia *POSIX* compliant, implementa con una segnatura diversa la funzione *shm_open* necessaria per creare la shared memory. Inoltre, come indicato dalla documentazione di Apple nella sezione bugs, *"The PTHREAD_PROCESS_SHARED attribute is not supported"*.

Entrambe queste situazioni rendono incompatibile la gestione del processo log, il resto del programma può essere compilato ed eseguito senza problemi.

3.2 Note Aggiuntive

1. **File di configurazione** - Se il file di configurazione non é settato correttamente, il server si arresta immediatamente.

2. **File nascosti** - I file nascosti, hanno il cui nome inizia con il carattere '.', quando richiesti il server invia al client un messaggio di errore.

3. **Windows Gestione richieste** Solo nella modalità processo, dopo creato il processo Handle_Request e avergli passato la socket clonata, attendiamo la sua terminazione con lo scopo di chiudere la socket (originale), perché le api di windows non permettono la chiusura della socket da parte del processo che la eredita.

4. **external ip address** È richiesta un hostname di dimensioni inferiori a 2048 byte.

5. **DOS_PROTECTION** La variabile DOS_PROTECTION viene utilizzata in caso di richieste superiori al limite imposto. Se uguale a 0, effettua un drain delle richieste. Altrimenti se il valore è diverso da 0 allora le richieste vengono chiuse immediatamente, il client riceverà un errore per via della prematura chiusura della socket. In entrambi i casi restituisce un messaggio di errore di tipo '3'.

Capitolo 4

Test Effettuati

4.1 Piattaforme

Il server è stato testato su distro Linux quali

1. Linux
   (a) ArchLinux
   (b) Manjaro
   (c) Debian 9
2. MacOS Mojave (senza gestione log)
3. Windows 10 ultima release stable (luglio 2019)

4.2 Client Testati

Il server è stato testato con diversi client gopher sia per windows che per linux, i test pricipali sono stati effettuati con *curl* e per questo lo riteniamo come client stabile.

4.3 Stress Test

Nella root del progetto è possibile trovare un file bash denominato 'stressCURL.sh' il quale lancia contemporaneamente un numero K di richieste verso il server.

Alcuni test effettuati per verificare la stabilità e la correttezza del server sono elencati di seguito.

1. **stressCurl.sh** - Sono state lanciate 50k richieste tramite stressCurl.sh per la ricezione di un file dal peso di 4Kb.
2. **Download Film** Abbiamo trasferito un file video da più di un gigabyte, la velocità di trasferimento si è attestata a circa la velocità del disco della macchina su cui è stato effettuato il test. Abbiamo successivamente verificato l'integrità del file ricevuto tramite il comando *cmp*.