SimpleHashr
Manuale

CERT-PA

February 26, 2020

Abstract

Hashr è uno strumento per la ricerca di evidenze[1] all'interno di una macchina[2]. Lo strumento, sviluppato dal CERT-PA, offre solo un'interfaccia da **linea di comando** (CLI). Lo sforzo necessario per sviluppare un'adeguata interfaccia grafica (GUI) non è giustificato dal ritorno in *usabilità* per il tipo di utenti a cui è destinato[3] Hashr.
Tuttavia un'importante fetta degli utilizzatori di Hashr non è abituato all'uso di una CLI e il supporto tecnico necessario che ne deriva non è trascurabile, per questo motivo è stata sviluppata una versione *minimale* di Hashr - detta **SimpleHashr** - che offre un'interfaccia grafica (GUI). SimpleHashr non è un *frontend* grafico o un prodotto derivato dal codice originale di Hashr, è un software indipendente che fa uso di tecnologie completamente diverse (es: Win32 C contro Python).
Oltre alla GUI SimpleHashr offre alcune funzionalità aggiuntive come una maggiore velocità di esecuzione, impiego di tutte le CPU disponibili, elevazione dei privilegi per la lettura dei file e l'assenza totale di dipendenze da componenti esterni[4].
Solo un limitato insieme delle funzionalità di Hashr è stato reimplementato, in particolare la ricerca di evidenze cifrate o in cartelle specifiche non è supportato.

1 Download ed installazione

SimpleHashr è scaricabile dalla sezione del sito del CERT-PA, che è solita contenere gli strumenti messi a disposizione dal Team operativo.

Qualora la necessità di usare SimpleHashr sia derivata da un'e-mail del CERT-PA, un link per il download è molto probabilmente incluso nel corpo della stessa.

1.1 32 o 64 bit

Essendo un'applicazione nativa SimpleHashr è disponibile nelle versioni a 32 e 64 bit.

Se si è incerti su cosa utilizzare, **scaricare la versione a 32 bit**.

Entrambe le versioni sono in grado di gestire file di *qualsiasi dimensione*[5] e le prestazioni sono simili, ma quella a 32 bit funziona anche su macchine obsolete (pre 2004).

Ovviamente la versione a 64 bit è più performante sul calcolo degli hash[6] e richiede meno overhead per file di grossissime dimensioni. Per questo motivo quest'ultima è vivamente consigliata.

1.2 Installazione

Non è necessaria installazione.

Fare *doppio click* sul file scaricato.

[1] File corrispondenti a specifici hash.
[2] Più nel dettaglio: all'interno di un *volume*.
[3] Principalmente tecnici.
[4] Ovviamente le DLL che compongono le API di Windows sono comunque presenti tra le dipendenze PE.
[5] In particolare la versione a 32 bit non alcun problema a gestire file superiori a 4GiB.
[6] In quanto l'artimetica a 64 bit è nativa.

2 Opzioni di ricerca

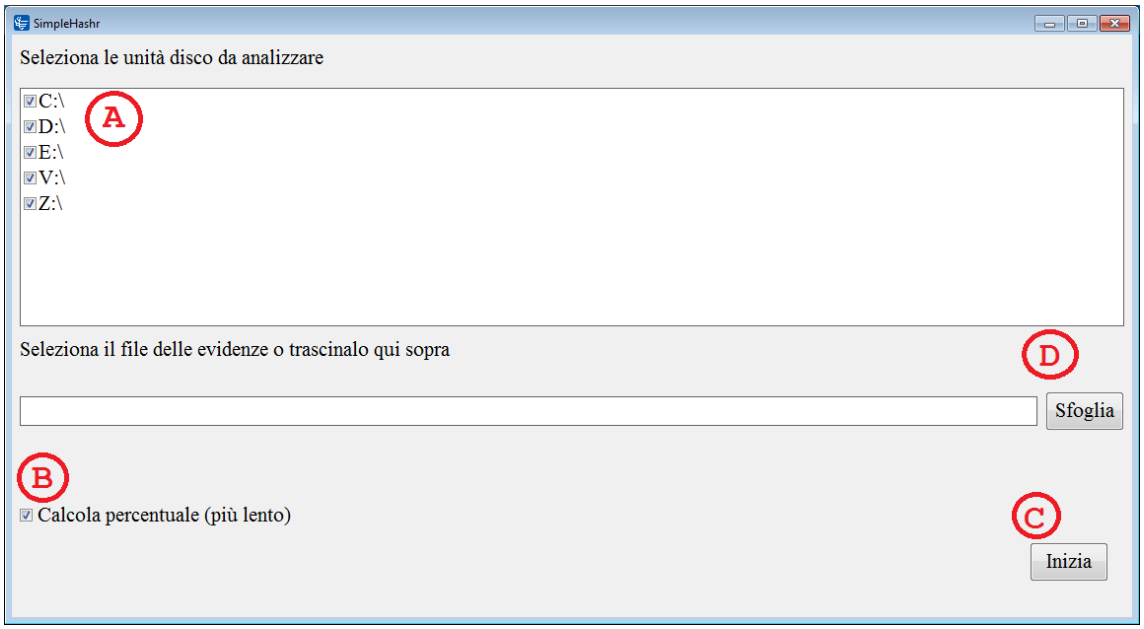

Figure 1: Opzioni di ricerca di SimpleHashr

2.1 Selezione dei dischi da controllare

Una volta aperto, SimpleHashr **enumererà tutti i dischi** (o meglio i volumi[7]) **accessibili dall'utente** e li mostrerà nella sezione contrassegnata in figura 1 con la lettera `A`.
Di default tutti i dischi accessibili sono selezionati, l'utente può deselezionarli o riselezionarli cliccando sul nome del disco o sulla spunta subito alla sua sinistra.
Al momento dell'avvio della ricerca, **solo i dischi selezionati saranno controllati**.

2.2 File delle evidenze

Come Hashr anche SimpleHashr necessita di un file contenente le evidenze da cercare. Questo file è tipicamente fornito insieme alla segnalazione.
Per scegliere il file delle evidenze è possibile usare il pulsante contrassegnato in figura 1 dalla lettera `D` (che mostra la classica finestra *Sfoglia* di Windows) o scrivere/incollare manualmente il percorso nella barra di testo subito alla sua sinistra.

2.3 Calcolo, *opzionale*, della percentuale

L'operazione di ricerca può essere piuttosto lunga, qualora lo si ritenga utile è possibile optare per visualizzare la percentuale di avanzamento.

Tuttavia affichè ciò sia possibile è necessaria una pre-scansione dei dischi[8], questa operazione è *molto* più veloce della ricerca delle evidenze ma in macchine particolarmente vecchie o con dischi poco performanti può richiedere svariati minuti[9].

Di default, il calcolo della percentuale è **abilitato** ma può essere disabilitato e riabilitato cliccando sulla spunta indicata in figura 1 con la lettera `B` o sul testo subito alla sua destra.

[7] In questo testo verrà usato il termine improprio "disco" per favorire gli utenti meno esperti.
[8] Nel dettaglio, è necessario enumerare tutti i file presenti nei volumi.
[9] 10 minuti è il massimo che abbiamo osservato. In una macchina moderna, richiede meno di 20 secondi.

3 Avvio, sospensione, ripresa, interruzione e completamento della ricerca

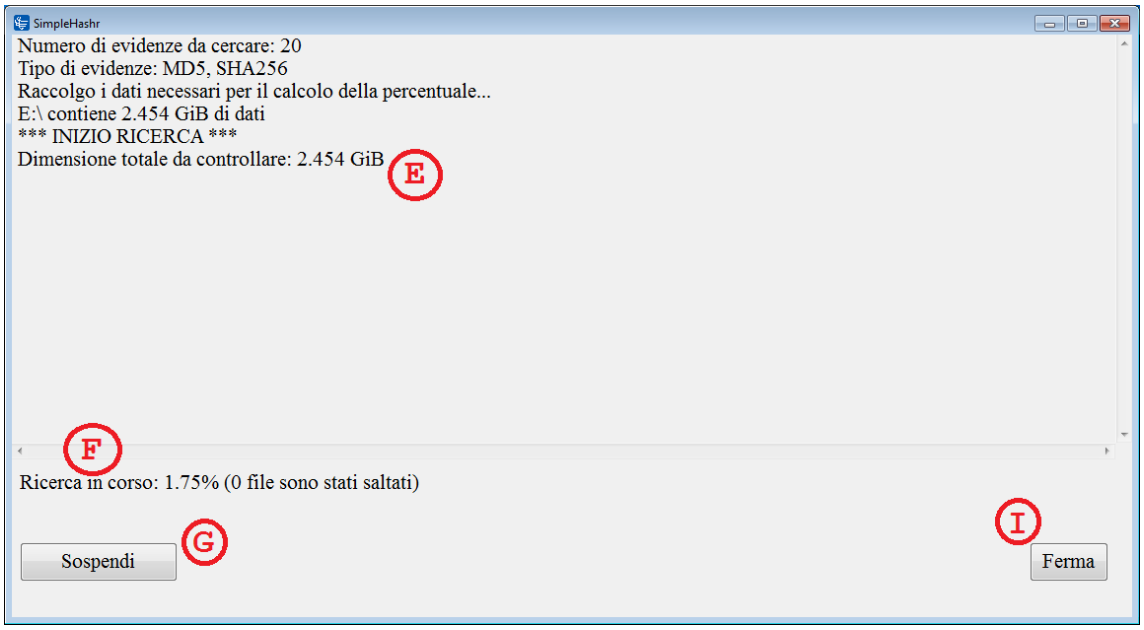

Figure 2: Ricerca in corso

3.1 Avvio

La ricerca si avvia tramite il pulsante indicato in figura 1 dalla lettera `C`.
Una volta avviata la ricerca può *essere messa in pausa* od *interrotta*.

L'avvio della ricerca *nasconde* la lista dei dischi e **mostra** un'area (indicata dalla lettera `E` in figura 2) in cui sono riportate informazioni rilevanti e **eventuali evidenze trovate**.
In basso, nel punto indicato in figura 2 dalla lettera `F`, viene mostrato l'avanzamento della ricerca con la percentuale completata (qualora sia stata scelta questa opzione) od uno degli ultimi file analizzati.

3.2 Sospensione

La ricerca può essere *sospesa* cliccando sul pulsante indicato in figura 2 dalla lettera `G`.
Quando la ricerca è sospesa SimpleHashr non occupa risorse computazionali[10] ma è comunque in grado di *riprendere* la ricerca da dove aveva lasciato.

La ricerca è sospesa il prima possibile ma non immediatamente, può passare un po' di tempo necessario a completare le singole analisi dei file già iniziate. Sospendere la ricerca permette di lavorare con comodità sulla macchina.

3.3 Ripresa

Una ricerca *sospesa* può essere *ripresa* cliccando sul pulsante indicato in figura 3 dalla lettera `H`.
Una volta ripresa la ricerca valgono le considerazioni della sezione Avvio.

3.4 Interruzione

Una ricerca, sia *sospesa* che *avviata*, può essere *interrotta* tramite il pulsante indicato in figura 2 dalla lettera `I`. L'interruzione **non è immediata**, il tempo necessario è solitamente basso e dipende

[10] La scansione dei file è messa in pausa e l'utilizzo della CPU e del disco è nullo.

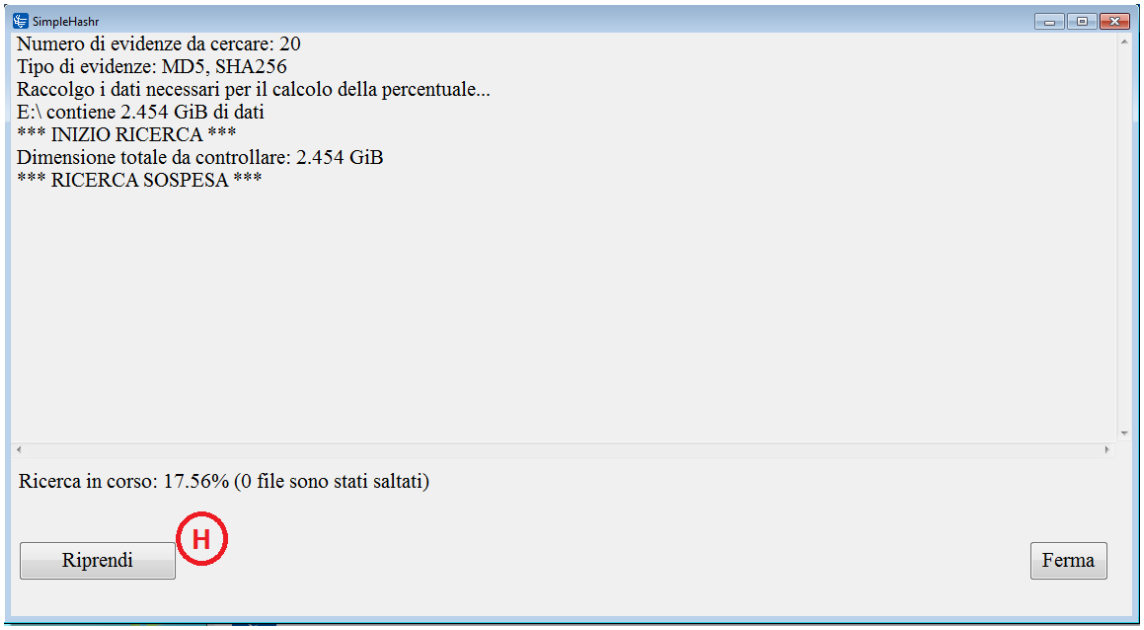

Figure 3: Ricerca sospesa

dal file in analisi.

ATTENZIONE Una volta interrotta una ricerca non può essere ripresa, è necessario **avviare una nuova ricerca** perdendo tutti i progressi fatti.

3.5 Completamento

Una volta completata una ricerca l'area indicata in figura 2 dalla lettera `E` conterrà le evidenze trovate. Nel punto indicato in figura 4 dalla lettera `L` verrà indicato il tempo che è stato impiegato per la ricerca.
Tramite il pulsante indicato in figura 4 dalla lettera `M` è possibile **chiudere l'area in cui sono riportate le evidenze** (lettera `D`) ed eventualmente iniziare una nuova ricerca.
Se al termine della ricerca non sono riportate evidenze, vuol dire che non ne è stata trovata alcuna.

ATTENZIONE Una volta chiusa l'area dove sono riportate le evidenze, **non sarà più possibile copiarle**[11].

4 Tempo di esecuzione

Il tempo impiegato da SimpleHashr per analizzare i dischi scelti dipende da tre fattori:

1. Dimensione dei dischi controllati.
2. Velocità di lettura dal disco.
3. Numero di CPU[12].

In misura minore anche l'indicizzazione dei file effettuata da Windows riduce i tempi di esecuzione.

[11] In realtà la finestra è solo nascosta ed il suo contenuto è cancellato all'avvio della prossima ricerca. Con un tool come *WinSpy* è possibile recuperare il testo.

[12] Non specifichiamo in questa sede la distinzione tra *thread (hardware)* e *core*. Questa è rilevante in base all'implementazione degli algoritmi di hashing. Versioni particolarmente ottimizzate non traggono vantaggio dall'esecuzione su thread fratelli (ovvero nel solito core) per via dell'utilizzo già ottimale delle risorse microarchitetturali.

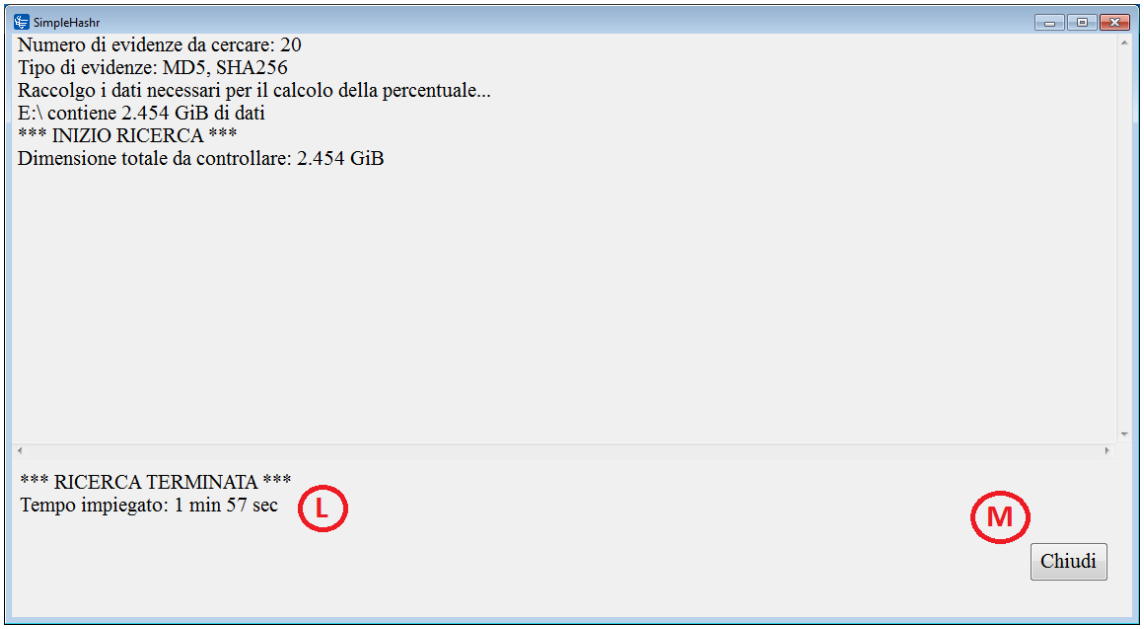

Figure 4: Ricerca completata

In generale, su macchine adeguate la ricerca si conclude nel giro di un'ora ma **è impossibile dare una stima valida per ogni situazione**. Nella nostra esperienza:

- **Su una macchina moderna** con 4 core WhiskeyLake ed un disco SSD SimpleHashr processa **50 GiB in 20 minuti**.
- **Su una macchina virtuale** con 2 core WhiskeyLake SimpleHashr ha impiegato **36 minuti per processare 54 GiB**.
- **Su una vecchia macchina** a 32 bit (architettura non meglio precisata) ed un disco particolarmente lento SimpleHashr ha impiegato più di **due ore per processare 30 GiB**.

5 Avanzate

Questa sezione contiene tematiche avanzate. Non è necessaria per l'uso di SimpleHashr.

5.1 Permessi, accessibilità e visibilità

SimpleHashr è un programma Windows, come ogni processo di Windows quando è lanciato gli viene assegnato il token dell'utente corrente. Tuttavia da Windows Vista in poi **i programmi lanciati dagli utenti amministratori non girano con un token amministrativo** a meno che questo non sia esplicitamente richiesto (tramite la finestra di elevazione UAC [13]).

Quindi SimpleHashr **non sempre ha i permessi di leggere alcuni file e cartelle** di Windows, o di altri utenti. Per ovviare è possibile lanciare SimpleHashr facendo tasto destro del mouse sull'eseguibile e scegliendo **"Esegui come amministratore"**.
Quando si esegue SimpleHashr con token con pieni privilegi amministrativi, i thread di ricerca si **elevano ai privilegi dell'account System locale**, questo permette loro di accedere a file normalmente non accessibili neanche agli amministratori.

Tuttavia Windows impedisce l'accesso in lettura ai file che sono già in uso da un'applicazione a meno che questa dia l'esplicito consenso, per cui alcuni file non sono leggibili per design di

[13] Programmaticamente tramite il verbo *runas* di `ShellOpenEx`, il token di un processo non è modificabile una volta assegnato.

Windows. Va inoltre aggiunto che un programma con sufficienti privilegi può negare la lettura di un file a tutti gli utenti, inclusi gli amministratori, è possibile ripristinare i permessi corretti ma SimpleHashr adotta la politica di non modificare *mai* i file analizzati.

Tutti i file saltati sono scritti nel file di log di nome *SimpleHashr.log* creato nella stessa cartella dell'eseguibile, se questo è possibile.

Infine, al di là dei permessi sui file, i volumi (dischi) possono essere montati **solo per alcuni utenti (ad esempio, questo è il caso delle cartelle condivise di VirtualBox**). Questo meccanismo è indipendente dai permessi ed è parte del design di Windows[14]: ogni utente ha il suo insieme di volumi. Lanciando SimpleHashr come amministratore e come utente normale è possibile che i volumi enumerati differiscano tra le due esecuzioni.

5.2 Memoria usata

La quantità di memoria usata da SimpleHashr è **piuttosto limitata**, in generale si attesta intorno ai 5-7 MiB.
Questa quantità **non dipende** dalla quantità e dimensione dei file analizzati, infatti SimpleHashr *non consuma memoria*[15] per l'analisi dei file ma solo per la gestione dello stato e dei metadati.
La versione a 32 bit mappa fino ad 1GiB per volta, quella a 64 bit fino a 256 GiB. Questi valori possono essere cambiati ricompilando SimpleHashr.

5.3 Sovraconteggio della percentuale

Il calcolo della percentuale avviene enumerando i file da controllare, una volta effettuata questa operazione i file sono di nuovo enumerati[16] e processati. Se tra le due enumerazioni vengono aggiunti nuovi file o nuovi dati ai file esistenti, la percentuale calcolata durante il progresso delle operazioni può superare il 100% di qualche punto.

5.4 Analisi live

Per tutte le considerazioni fatte sopra riguardo ai permessi, file in uso e modifiche dei file, si **consiglia l'analisi su una copia della macchina**.
Qualora, per risparmiare tempo, si proceda ad un'analisi su una macchina attiva, si **consiglia** di revisione il file log (presente nella stessa cartella dell'eseguibile) creato da SimpleHashr per verificare la presenza di file sospetti tra quelli saltati[17].

6 Architettura

6.1 Sorgenti

I sorgenti sono disponibili sul sito del CERT-PA (si veda le rilevanti considerazioni fatte in Download ed installazione.

SimpleHashr è scritto in C11 ma è compatibile con C99 ed utilizza le API Win32[18], può quindi essere compilato con un qualsiasi compilatore C11 a patto di avere a disposizione gli headers di Windows[19].
Analogamente, per il linking è necessario avere a disposizione i file con le definizioni delle funzioni delle varie librerie (`.lib`) ed un linker che produce un output *PE* o *PE32+*.

[14] É possibile far impersonare ogni utente disponibile sul sistema ai thread di ricerca ma questo non è stato fatto.
[15] I file sono *mappati* in memoria, non letti.
[16] Salvarsi la lista avrebbe un costo troppo oneroso in termini di memoria.
[17] **Moltissimi** malware cambiano i permessi di lettura dei propri file per renderli inaccessibili ad altri utenti. SimpleHashr non può sapere se il file è malevolo o meno e non prova a diventare proprietario del file in quanto questo può compromettere l'installazione di Windows se fatto sui file di sistema.
[18] Ovvero le uniche API di Windows native, se si considera che tecnologie come WinRT sono comunque basate su Win32. Il numero 32 non deve far pensare a 32 bit, è lo stacco che c'è stato tra Win16 e Win32, il nome include anche la parte a 64 bit delle API.
[19] Che si trovano anche nell'SDK relativo.

Nello specifico SimpleHashr è stato sviluppato con Dev-C++, un IDE **obsoleto** che usa MinGW come ambiente di compilazione, il suo vantaggio è quello di essere *molto* più leggero di Visual Studio Express (lo strumento de facto usato per lo sviluppo nativo) ma è molto più rudimentale.

6.2 Dati per la percentuale

Quando richiesto di mostrare la percentuale di progresso SimpleHashr effettua una prima scansione dei volumi in cui colleziona la *dimensione totale* dei dati da analizzare.
Questa scansione è generalmente veloce, se si ha attiva l'indicizzazione dei file, poichè non è effettuato nessuno IO su disco da parte di SimpleHashr (se si esclude quello effettuato da `FindFirstFile` e `FindNextFile`).

Per ogni volume scelto SimpleHashr avvia un thread in background che enumera tutti i file nel volume e colleziona la **dimensione totale** di questi. Le dimensioni totali dei singoli volumi sono poi sommate insieme[20] per ottenere la dimensione globale dei dati da controllare.
SimpleHashr usa quindi la dimensione dei file (e non il *numero* di file) processati come indicatore di progresso.

6.3 Processo di ricerca

La ricerca avviene in modo simile alla raccolta dei dati per la percentuale: per ogni volume SimpleHashr avvia un thread (di ricerca), il quale enumera tutti i file nel volume e li aggiunge ad una coda *globale* (la cui dimensione massima è di 128 elementi).
Inoltre SimpleHashr avvia un thread (di lavoro) per ogni *cpu logica*, questo thread rimuove un elemento alla volta dalla coda e ne calcola gli hash necessari (solo i tipi di hash necessari sono calcolati). I thread di lavoro riportano la dimensione dei dai processati ed aggiornano, eventualmente, la percentuale fatta.

Quando un thread di ricerca esce, controlla se è l'ultimo rimasto ed in caso positivo inserisce un valore speciale nella coda che indica la fine del lavoro. Dopodiché questo ultimo thread aspetta che i thread di lavoro terminino.
Quest'ultimi, quando vedono il valore speciale nella coda, capiscono che non vi sono più file da processare e terminano. L'ultimo thread di lavoro a terminare registra il tempo impiegato per l'analisi e notifica all'unico thread di ricerca rimasto l'avvenuta fine dell'analisi. Una volta ottenuta questa notifica, il thread di ricerca comunica alla GUI la fine delle operazioni.

6.4 Ricompilazione

Gli utenti programmatori possono trovare utile ricompilare SimpleHashr. Tecnicamente parlando, ricompilare SimpleHashr *per la propria architettura* permette di ottenere un incremento (anche notevole) in termini di prestazioni poichè sarà possibile usare tutte le estensioni x86 disponibili.

6.4.1 Range mappato

SimpleHashr mappa i file un "segmento"[21] alla volta, nella versione a 32 bit questo ha dimensione 1 GiB, in quella a 64 bit ha dimensione 256 GiB.
Il "segmento" è anche l'unità atomica di analisi, ovvero quando richiesto di interrompere la ricerca SimpleHashr porterà comunque a termine l'analisi del segmento in corso.

Qualora lo si ritenesse necessario è possibile cambiare la dimensione del "segmento" nel file `simplehashr.h`.

[20] In una sezione critica.
[21] Qui "segmento" è usato in modo informale, non tecnico. Nessun riferimento all'architettura x86 o ad unità di misura.

6.4.2 Estensioni x86

Il motivo più valido per cui ricompilare SimpleHashr è senza dubbio la possibilità di usare istruzioni specifiche (dato che Intel e AMD fanno sforzi notevoli proprio a riguardo).
SimpleHashr è compilato per un'architettura generica, la baseline per le versione a 64 bit include comunque *SSE 2* ma quella a 32 bit può non far uso di questa estensione.
É molto probabile, se la macchina non è troppo vecchia, che si abbia a disposizione almeno *AVX*, questo **insieme ad un'implementazione degli hash apposita** (si veda *gcrypt*) può notevolmente abbassare il tempo di calcolo necessario[22].

6.4.3 Coda dei file

La comunicazione tra i thread di ricerca e quelli di lavoro avviene tramite una coda di massimo 128 elementi. Questo significa che in ogni momento SimpleHashr non avrà mai più di 128 file aperti contemporaneamente.
Se i file da processare sono molto grossi i thread di lavoro arrivano velocemente a saturazione della coda, restando poi in attesa che si liberi un posto. Una coda più lunga aiuterebbe a nascondere la latenza dei thread di lavoro a beneficio del throughput (ma non della latenza globale di un singolo file!) dei thread di ricerca.
Una coda più lunga comporta un maggior dispendio di memoria (circa 4KiB ad elemento) ed il rischio di raggiungere il numero massimo di `HANDLE` disponibili. Questo, insieme al fatto che l'operazione di enumerazione dei file è molto veloce (se comparata con il calcolo degli hash), porta ad avere pochi motivi per incrementare la lunghezza della coda.

[22] Attenzione, sui server, all'uso di AVX-512. Queste istruzioni comportano un passaggio alla licenza di frequenza più bassa (L2), mentre si ha comunque un vantaggio nell'usarle, **tutto** il software ne risente, inclusi eventuali web server che risulterebbero rallentati. Ciò è vero anche per AVX2 in parte, ma in misura molto minore.