



**UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO**

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA IN SICUREZZA INFORMATICA

**Tesi di Laurea in
INFORMATICA FORENSE**

**REALIZZAZIONE DELL'INSTALLER PER LA DISTRO
FORENSE TINY CORE FORENSIC EDITION**

Relatore:

Prof. Ugo LOPEZ

Laureando:

Flavio COLLOCOLA

Anno Accademico 2022-2023

Ringraziamenti

Ringrazio nonna Lina per essere arrivato fin qui. Con la sua dolcezza e gentilezza ha sempre creduto in me e in tutti noi nipoti. Ci ha sempre voluto bene e l'ha sempre dimostrato in tutti i modi, non si poteva avere una nonna migliore di lei!

Ringrazio i miei genitori per aver creduto nel mio percorso e aver sostenuto i miei studi. Ringrazio anche mio fratello, che nonostante sia un capacchione, gli voglio comunque bene.

Ringrazio Valentina, che nel momento più inaspettato è comparsa nella mia vita. Una delle ragazze più dolci, solari e simpatiche che abbia mai conosciuto, non avrei mai detto che fosse uno "Scorpione". Il tempo vola quando stai in sua compagnia: una chiamata al telefono di due ore e mezza non sembra essere durata nemmeno 20 minuti. Se vai al bar con lei, lo Spritz è d'obbligo. E a proposito di bar... non posso non ringraziarla anche per tutti i dolcini che mi ha ceduto!

Ringrazio il mitico zio Guido, migliore amico e collega ormai dal primo anno di università (sì, mi sopporta da ben 5 anni). Lo zio Guido è quell'amico che tutti vorrebbero, non solo per la sua gentilezza e disponibilità, ma anche per il suo ampio repertorio di meme (che non so dove li trova). Se vuoi farti una birra in compagnia, lui c'è sempre, che siano le 3 di pomeriggio o le 3 di notte. C'è però una cosa che proprio non sopporto di lui... quando vince le partite di biliardo all'ultimo secondo!

Ringrazio Angelo, altro mio migliore amico. Con Angelo il divertimento è assicurato, dalle risate a crepappe... ai balli di gruppo. Per non parlare poi delle serate in cui sta a Shazammarsi tutte le canzoni che metto in macchina o le volte in cui gli faccio ascoltare in anteprima i brani che compongo. Andare a studiare alla sede di Giurisprudenza è uno spasso quando c'è lui (infatti si finisce per non studiare). Chissà se un giorno comprerà il bar di fronte all'università...

Ringrazio Alberto e Grace, due grandi amici. Alberto porta con sé uno stile di vita, l'Albertismo. Con questo stile di vita riesce ad Albertizzare tutti. Uno degli esempi più palesi è dato dal fatto che nomina chiunque collega o nemico, e ci fa una canzoncina intonata con il nome della persona, ecco, da quando l'ho conosciuto anche io chiamo tutti colleghi o nemici e ci faccio la canzoncina col nome. Alla collega Grace (ecco nemmeno a farlo apposta), gli abbiamo fatto tanto di testa con queste canzoncine, ma nonostante tutto ancora ci sopporta. Sicuramente la collega Grace ne avrà scritte tante su di noi nel suo romanzo... comprendo la sua pazienza.

Ringrazio il mio relatore, il professor Ugo Lopez, per la sua disponibilità, professionalità e pazienza nel seguirmi su questo progetto di tesi. Ringrazio inoltre Marilena Porfido e Michele Troccoli per l'esperienza formativa di tirocinio, in cui non mi sarei mai aspettato mai di provare l'ebbrezza di insegnare o di svolgere indagini forensi in contesti reali.

Ringrazio i miei amici e colleghi universitari tra cui Fram, con le sue idee e progetti geniali, Andrea, con le sue abilità da Personal Trainer, e Claudio, collega fisso nei progetti universitari.

Introduzione

Ringrazio Vittorio, il miglior guardiano che la nostra sede universitaria possa avere. Le pause durante le lezioni erano uno spasso quando era di turno, non potevi non farti una lunga e interessante chiacchierata con lui.

Sommario

1. <i>Introduzione</i>	7
1.1. Le scienze forensi	7
1.2. Storia dell'informatica forense	8
1.3. Rami della digital forensics	11
1.4. Aspetti legali	12
2. <i>Stato dell'arte</i>	13
2.1. Distro forensi	13
2.1.1. CAINE	14
2.1.2. Tsurugi	15
2.1.3. Tiny Core Forensic Edition	16
3. <i>Analisi e progettazione</i>	19
3.1. Definizione del problema	19
3.2. Aggiornamento del Core	20
3.2.1. Aggiornamento di Volatility 3	20
3.3. Tiny Core Forensic Edition Installer	22
3.3.1. Requisiti	23
3.3.2. Interfaccia grafica	23
4. <i>Implementazione</i>	27
4.1. Prerequisiti	27
4.1.1. Compilazione di termcap	29
4.1.2. Compilazione di GNU Parted	31
4.1.3. Python3.9 pyparted	34
4.1.4. Python3.9 tk	35
4.1.5. Python3.9 pillow	36
4.2. Realizzazione dell'interfaccia grafica	37
4.3. Analisi del codice	38
4.3.1. Gestione delle partizioni	39
4.3.2. Copia dei file	40
4.3.3. Installazione del bootloader	41
4.3.4. Script di supporto dell'installer	44

4.3.5.	Startup script.....	45
4.4.	Packaging dell'installer	45
5.	Creazione della ISO	46
5.1.	Personalizzazione del Core per la modalità Live	46
5.2.	Personalizzazione del Core per la modalità installazione	47
5.3.	EzRemaster.....	49
5.4.	Personalizzazione di ISOLINUX	55
5.5.	Script per la creazione della ISO finale (makeiso.sh)	57
6.	Testing.....	58
6.1.	Esecuzione della modalità Live	59
6.2.	Esecuzione dell'installer	60
6.2.1.	Test 1: Partizionamento automatico	60
6.2.2.	Test 2: Partizionamento manuale.....	63
7.	Risultati.....	68
8.	Sviluppi futuri.....	68
9.	Bibliografia.....	69

1. Introduzione

Nell'accezione tradizionale, il termine “forense”, indica sia una forma di evidenza giuridica sia una categoria di pubblicizzazione legale.¹

Difatti, il termine deriva dal latino “forum” (fòro, piazza), ad indicare che, quando nell'antica Roma si processava qualcuno per un reato, il caso veniva esposto pubblicamente nel forum. In tal modo sia l'imputato che il Pubblico Ministero potevano fornire le loro evidenze.

Il significato di tale termine subisce un'accezione differente nell'età moderna, ovvero agli aspetti legali si lega strettamente il concetto di scienza. D'ora in poi, infatti, verranno applicate tecniche e metodologie scientifiche alle tradizionali investigazioni di carattere giudiziario.

1.1. Le scienze forensi

Le scienze forensi sono una disciplina abbastanza recente. Essa consiste nell'applicazione delle tecniche e metodologie scientifiche alle investigazioni di carattere giudiziario, al fine di accertare un reato o un comportamento sociale.

Le scienze forensi, inoltre, sono suddivise in molteplici branche, a seconda del settore di applicazione. Tra queste si annoverano l'antropologia forense, l'archeologia forense, la balistica forense, la chimica forense, la dattiloscopia forense, la medicina forense, la dattiloscopia forense, ecc...

La paternità di tali scienze è attribuita ad Alphonse Bertillon, un criminologo che lavorava in una questura francese e che, nel 1870, fondò il primo laboratorio di identificazione criminale e inventò un sistema di riconoscimento biometrico (sistema Bertillon) successivamente adottato sia in Europa che negli Stati Uniti.²

¹ https://it.wikipedia.org/wiki/Scienza_forense

² https://en.wikipedia.org/wiki/Alphonse_Bertillon

In generale, alla base delle scienze forensi vi è il principio di Locard (in onore del criminologo francese del XIX secolo Edmond Locard) secondo cui “Non si può entrare/uscire da un luogo senza lasciare qualcosa di sé”.

Nel 1880, Henry Faulds (medico e missionario in Giappone), pubblicò un articolo sulla rivista scientifica *Nature* in cui illustrò l'univocità delle impronte digitali e la possibile applicazione in campo forense. Analogamente alle impronte digitali, anche il DNA è utilizzato in ambito forense.

L'analisi forense del DNA fu sviluppata da Sir Alec Jeffreys e utilizzata per la prima volta nel 1984. Il principio alla base dell'analisi forense del DNA è il fatto che le variazioni nella sequenza del DNA sono in grado di identificare univocamente gli esseri umani.

Data la validità e l'importanza di questa metodologia di analisi, sono stati sviluppati database dei DNA nazionali e internazionali. Questi database sono utilizzati per confrontare le tracce di DNA rilevate sulle scene del crimine con DNA di criminali noti.

In questo contesto si colloca anche la moderna informatica forense, in cui si recuperano e analizzano prove derivanti da svariati dispositivi digitali.

Visto, inoltre, l'enorme sviluppo e utilizzo dell'informatica nel vivere quotidiano, l'informatica forense costituisce, al giorno d'oggi, uno dei principali mezzi investigativi.

1.2. Storia dell'informatica forense

Sebbene la scienza forense è già relativamente una disciplina recente, l'informatica forense lo è ancora di più. Essa infatti si sviluppa a partire dagli anni '80 del 1900, periodo in cui si iniziarono a diffondere i Personal Computer tra le masse.³

³ https://it.wikipedia.org/wiki/Informatica_forense

In particolare, nel 1984, l'FBI e altre agenzie investigative iniziano a sviluppare software per rilevare indizi all'interno dei PC. In tale anno, l'FBI creò anche il Computer Analysis and Response Team (CART), un team oggi composto da 500 agenti esperti e certificati nel campo dell'analisi forense. Essi analizzano svariati supporti digitali sequestrati come computer, smartphone, fotocamere e lettori multimediali, memorie, CD/DVD, ecc...⁴

Una prima diffusione di Internet negli anni '90 aumentò notevolmente la quantità di crimini informatici, rendendo molto più attiva tale disciplina. In questo periodo infatti vengono sviluppati numerosi software forensi a supporto delle indagini riguardanti crimini informatici. Tra i primi software sviluppati si annovera IMDUMP (sviluppato nel 1989 da Michael White), SafeBack (sviluppato da Sydex) e The Coroner's Toolkit (TCT). Sia IMDUMP che SafeBack servivano per acquisire immagini bit stream forensi dai dispositivi di archiviazione, il The Coroner's Toolkit invece era una collection di software forensi e di sicurezza informatica open source, in grado di funzionare su sistemi operativi Unix based.

Tuttavia, solo negli anni 2000, l'informatica forense e le procedure e metodologie che ne fanno parte, subiscono un processo di standardizzazione. Nel 2002, il Scientific Working Group on Digital Evidence (SWGDE) produsse un documento dal titolo "Best practice for Computer Forensics". Nel 2005, tale documento fu seguito dalla pubblicazione dello standard ISO 17025 (General Requirements for the Competence of Testing and Calibration Laboratories).

Un ulteriore passo avanti in tal campo fu fatto nel 2004, periodo in cui entrò in vigore la "Convenzione sulla criminalità informatica". L'obiettivo di tale convenzione fu quello di raggiungere un accordo a livello internazionale

⁴ <https://www.fbi.gov/news/stories/piecing-together-digital-evidence>

sulla criminalità informatica, perseguendo una politica comune di contrasto ai crimini informatici, definendo tecniche investigative comuni e favorendo la cooperazione tra Stati.

Infine, l'esplosione della tecnologia e di Internet avvenuto dal 2007 in poi, ha introdotto numerose difficoltà nel campo dell'informatica forense. Principalmente, queste difficoltà sono dovute all'introduzione della crittografia nei dispositivi commerciali (come smartphone, tablet, PC, ecc...), l'incremento esponenziale della capacità di archiviazione dei dispositivi e la diffusione di un'ingente quantità di dispositivi tecnologici pro capite. Allo stesso tempo anche la delocalizzazione delle risorse e delle informazioni personali (dovuta alla diffusione del Cloud e dei servizi remoti), ha impattato in maniera non poco rilevante sulla digital forensics. Recentemente, il crescente numero di attacchi informatici (incentivato dall'utilizzo e alla migrazione dei servizi in Internet) ha causato numerose violazioni della privacy, talvolta di ingente entità. Tutto ciò ha spostato il focus sulla ricerca e raccolta di prove riguardanti attacchi informatici e violazioni della privacy.

1.3. Rami della digital forensics

La vastità di device, software, sistemi operativi, codifiche, tipi di dati, nonché attacchi informatici, ha portato alla settorizzazione dell'informatica forense:

- **Computer Forensics:** si occupa dell'identificazione, acquisizione e analisi di indizi digitali presenti su computer, sistemi embedded e memorie statiche come pendrive USB, hard disk meccanici e SSD, ecc... All'interno della Computer forensics sono presenti dei sottorami quali la Windows Forensics, Ubuntu Forensics, ecc...;
- **Mobile Forensics:** si occupa dell'identificazione, acquisizione, analisi e recupero di indizi digitali presenti su dispositivi portatili come smartphone, tablet, palmari e simili. All'interno della Mobile Forensics sono presenti la Android Forensics e la iPhone Forensics;
- **Network Forensics:** si occupa dell'identificazione, acquisizione e analisi del traffico di rete tra vari dispositivi collegati sia in reti locali e che in WAN, al fine di rilevare eventuali indizi digitali;
- **Database Forensics:** si occupa dell'identificazione, analisi e acquisizione di dati e metadati presenti all'interno di database al fine di recuperare eventuali indizi digitali;
- **Cloud Forensics:** è una branca che si è sviluppata recentemente, a seguito dell'avvento del Cloud Computing, e si occupa dell'identificazione, analisi e acquisizione di dati presenti in infrastrutture Cloud;
- **IoT Forensics:** come la Cloud Forensics, la IoT Forensics è una branca relativamente nuova dell'informatica forense ed ha come obiettivo identificare, analizzare e acquisire dati e metadati dai dispositivi IoT;

- **Bitcoin Forensics:** si occupa dell'identificazione, acquisizione e analisi dei dati presenti in un sistema di criptovalute.

1.4. Aspetti legali

Vista la mole di contenuti sensibili presenti nei dispositivi digitali, l'analisi di tali dispositivi, nonché dei dati presenti al loro interno è regolamentata a livello nazionale e internazionale.

In Italia, i primi progressi in tale ambito si ebbero con la legge n. 547/1993. Tale legge apportò le seguenti novità:

- Introduzione del concetto penalistico di sistemi informatici e telematici nonché di applicativi per elaboratore;
- Introduzione del concetto di documento informatico e domicilio digitale;
- Attribuzione di rilevanza penale ad azioni illegittime sulle password;
- Introduzione del concetto di malware;
- Introduzione del concetto di corrispondenza informatica e comunicazioni informatiche;
- Estensione della truffa anche a frode informatica.

Il 18 marzo 2008 venne promulgata la legge n. 48/08, la quale definisce come utilizzare in tribunale le prove ottenute da un'indagine forense. Tale legge ha ratificato la Convenzione del Consiglio d'Europa sulla criminalità informatica. In sintesi, la norma prevede: ⁵

- L'introduzione del concetto di frode relativa ai servizi di posta elettronica;
- L'estensione del concetto di danneggiamento ai sistemi informatici e telematici;
- Sanzioni più pesanti per i reati informatici;

⁵ https://it.wikipedia.org/wiki/Informatica_forense

- Norme più efficienti per contrastare la pedopornografia;
- Norme sulla perquisizione e il sequestro di dati informatici;
- Maggiori tutele per i dati personali;
- Introduzione del “Fondo per il contrasto della pedopornografia su Internet e per la protezione delle infrastrutture informatiche di interesse nazionale”.

2. Stato dell'arte

Attualmente, gran parte del software dedicato all'analisi forense è open source ed è progettato per l'esecuzione sia in ambiente Linux che Windows. Tuttavia, oltre a programmi specifici dedicati all'analisi forense (come il The Coroner's Toolkit (TCT), Autopsy e simili), sono stati sviluppati degli interi sistemi operativi (open source e non) su base Linux, al cui interno sono raccolti numerosi tool e utility dedicati alla digital forensics. Tali sistemi operativi, oltre a mettere a disposizione vaste raccolte di software forense, includono anche driver e configurazioni in grado di non alterare i reperti digitali in fase di analisi (es. Write Blocker).

2.1. Distro forensi

Come anticipato nel paragrafo precedente, esistono diverse distribuzioni basate su Linux dedicate all'analisi forense. Di queste, alcune sono in stato di abbandono (come DEFT e Iritaly), mentre altre vengono regolarmente aggiornate e sono utilizzate nelle indagini forensi.⁶

⁶ <https://www.cybersecurity360.it/soluzioni-aziendali/distribuzioni-forensi-per-linux-a-cosa-servono-e-le-migliori-per-le-analisi-forensi/>

2.1.1. CAINE

CAINE Linux è una distribuzione italiana Linux Live, basata su Ubuntu. Il nome della distribuzione è l'acronimo di Computer Aided INvestigative Environment.⁷

CAINE Linux nacque nel 2008 da un progetto di tesi dello studente Giancarlo Giustini, ed è oggi mantenuta dal dott. Nanni Bassetti.⁸

CAINE Linux fornisce numerosi tool forensi ed è dotata di un'interfaccia grafica molto semplice da utilizzare. Tra i tool forensi inclusi, si annoverano:

- **Autopsy:** una piattaforma open source dedicata all'analisi forense, dotata di interfaccia grafica e basata su The Sleuth Kit e altri software forensi;
- **Fsstat:** un software in grado di mostrare i dettagli associati al file system presente su un dispositivo di archiviazione;
- **PhotoRec:** un software in grado di recuperare i file persi da hard disk, dischi ottici e fotocamere digitali;
- **RegRipper:** un tool open source in grado di estrarre informazioni da file di Registro;
- **The Sleuth Kit:** un insieme di tool open source in grado di effettuare l'analisi forense di dischi e file system;
- **Wireshark:** un software in grado di monitorare e analizzare il traffico di rete.

L'attuale versione di CAINE Linux è la 13.0 "Warp" (basata su Ubuntu 22.04) che introduce un Write Blocker migliorato che blocca automaticamente la scrittura sui dischi rigidi.

⁷ https://en.wikipedia.org/wiki/CAINE_Linux

⁸ <https://www.caine-live.net>

2.1.2. Tsurugi

Tsurugi è una distribuzione Linux-based, altamente personalizzata al fine di fornire un kit completo di strumenti per l'analisi forense, la malware analysis e le attività di OSINT.⁹

Si tratta di una distribuzione tutta italiana, nata nel 2018 e sviluppata da Giovanni Rettare e Marco Giorgi e presentata in Giappone durante la conferenza AvTokio. Il team di sviluppo di Tsurugi è attualmente composto dagli sviluppatori di BackTrack Linux e di DEFT Linux (distro forense ormai abbandonata).

Come CAINE Linux, anche Tsurugi è basato su Ubuntu, in particolare l'ultima versione è anch'essa basata su Ubuntu 22.04 LTS: la versione LTS assicura un supporto prolungato per quanto riguarda gli aggiornamenti di sicurezza e le correzioni di bug.

Anche Tsurugi è avviabile in modalità Live (in tale modalità non si altera lo stato del computer in cui viene eseguito, mantenendo inalterate eventuali prove) o è installabile su PC fisici o su macchina virtuale. Come ogni distribuzione forense, Tsurugi include una raccolta di software dedicati all'acquisizione, analisi ed altre attività forensi.

Sono disponibili tre versioni di Tsurugi:

- **Tsurugi Linux [LAB]:** è la distribuzione Linux vera e propria, contenente svariati tool a supporto dell'acquisizione e analisi forense. L'ultima versione si basa su Ubuntu 22.04 LTS (64-bit con kernel Linux 6.0.2 custom). Tsurugi Linux [LAB] è disponibile sia come ISO, sia come macchina virtuale preconfigurata;
- **Tsurugi Acquire:** è la versione light di Tsurugi Linux [LAB], che include esclusivamente i tool dedicati all'acquisizione forense. A

⁹ <https://tsurugi-linux.org>

differenza di Tsurugi Linux [LAB] non si basa su Ubuntu, ma su Debian 10 (32-bit con kernel Linux 5.11.6): questa scelta è dovuta alla necessità di eseguire Tsurugi Acquire anche su hardware datato.

Inoltre, Tsurugi Acquire può essere eseguito solo in modalità Live ed è disponibile solo come ISO;

- **Bento:** a differenza dei precedenti, Bento non è un sistema operativo vero e proprio, ma è una raccolta di software forensi per Windows, Linux e macOS, utilizzabili, ad esempio, durante l'analisi forense sul campo.

2.1.3. Tiny Core Forensic Edition

Tiny Core Forensic Edition è una distribuzione forense piuttosto leggera, basata su Tiny Core Linux. Il sistema ha un tempo di avvio molto ridotto e il core è eseguito interamente in RAM.

L'idea è nata da un progetto di tesi dell'Università degli Studi di Bari proposto da Ruffo Sara. In seguito, il progetto è stato esteso grazie ad altri tesisti quali De Benedittis Francesca, che ha aggiunto funzionalità avanzate come un Write Blocker e al tesista Oliva Angelo, che ha realizzato il porting di Tiny Core Forensic Edition sul RaspberryPi, dando vita a piCore Forensic Edition.

Tiny Core Forensic Edition include i seguenti software:

- **BlockDev:** è il Write blocker di Tiny Core Forensic Edition. Si integra a livello di kernel e non è irreversibile. La gestione del blocco/sblocco delle unità è delegata al software BlockDev Gui;¹⁰
- **BlockDev Gui:** è il gestore grafico di BlockDev. BlockDev Gui fornisce un'interfaccia grafica piuttosto semplice dalla quale è

¹⁰ De Benedittis F. - Realizzazione di funzionalità avanzate in Tiny Core Forensic Edition

possibile attivare o disattivare il Write Blocker per ogni unità presente sul computer, nonché monitorarne lo stato;

- **Dd:** è una utility da linea di comando integrata nei sistemi operativi Unix based che ha principalmente lo scopo di copiare e convertire file. In particolare, il tool è in grado di copiare i dati in blocchi, bit per bit e di convertire i dati utilizzando codifiche differenti;¹¹
- **DdRescue:** è un software che consente il recupero dei dati. In particolare permette la copia di dati da un file o dispositivo a blocchi ad un altro cercando di recuperare il contenuto originale in caso di errori di lettura;¹²
- **Ewfacquire:** è una utility in grado di acquisire il contenuto di dispositivi di archiviazione e memorizzarlo nel formato EWF (*Expert Witness Compression Format*). Ewfacquire può anche convertire un immagine (dd) pura nel formato EWF [1];¹³
- **Foremost:** è un tool di recupero dati forense in grado di effettuare il file carving, ovvero recuperare i file basandosi su parti di essi come header, footers e strutture dati. Nonostante sia stato sviluppato principalmente per le forze dell'ordine, il codice sorgente e il software compilato sono disponibili gratuitamente per il download;¹⁴
- **Nmap:** è un software open source creato per effettuare scansioni di rete e security auditing. In particolare, il tool è in grado di scoprire informazioni sui servizi e sugli host in una rete di computer. Nmap è inoltre estensibile tramite script, che espandono notevolmente le funzionalità di questo software;¹⁵

¹¹ <https://www.geeksforgeeks.org/dd-command-linux/>

¹² https://www.gnu.org/software/ddrescue/ddrescue_it.html;

¹³ <https://linux.die.net/man/1/ewfacquire>

¹⁴ [https://en.wikipedia.org/wiki/Foremost_\(software\)](https://en.wikipedia.org/wiki/Foremost_(software))

¹⁵ <https://it.wikipedia.org/wiki/Nmap>

- **Netcat:** è un programma a riga di comando per la comunicazione remota. Esso può funzionare sia su protocollo TCP che UDP;¹⁶
- **Rhash:** acronimo di Recursive hash, è un tool a riga di comando per calcolare e verificare gli hash dei file. Il tool supporta i seguenti algoritmi di hash: CRC32, CRC32C, MD4, MD5, SHA1, SHA256, SHA512, SHA3, AICH, ED2K, DC++ TTH, BitTorrent BTIH, Tiger, GOST R 34.11-94, GOST R 34.11-2012, RIPEMD-160, HAS-160, EDON-R, e Whirlpool;¹⁷
- **SharePoint Forensic Downloader:** è un tool grafico scritto in Python destinato all'acquisizione forense di dati e metadati presenti sul servizio Microsoft SharePoint;¹⁸
- **Sleuthkit:** è una raccolta di tool forensi disponibili sia per Windows che per Unix, destinati all'estrazione di dati dai dischi rigidi e da altri dispositivi di archiviazione, al fine di facilitare l'analisi forense dei sistemi informatici. Sleuthkit è alla base del noto software forense Autopsy;¹⁹
- **Volatility3:** è uno dei più avanzati e popolari framework per il recupero di dati dalle memorie volatili come le RAM. Il framework non usa le API native dei vari sistemi operativi, ma funziona in maniera del tutto indipendente dal sistema operativo in esecuzione ed è in grado di mostrarne lo stato a runtime.²⁰

L'attuale versione di Tiny Core Forensic Edition tuttavia non dispone di un installer, ma è esclusivamente eseguibile in modalità Live. Inoltre tale versione è obsoleta, in quanto da un lato utilizza una vecchia versione del

¹⁶ <https://it.wikipedia.org/wiki/Netcat>

¹⁷ <https://github.com/rhash/RHash>

¹⁸ De Benedittis F. - Realizzazione di funzionalità avanzate in Tiny Core Forensic Edition

¹⁹ https://en.wikipedia.org/wiki/The_Sleuth_Kit

²⁰ <https://github.com/volatilityfoundation/volatility3>

kernel Linux e dall'altro lato i tool presenti all'interno non sono aggiornati alle ultime versioni disponibili in rete.

3. Analisi e progettazione

Tiny Core Forensic Edition è una distribuzione basata su Linux, affidabile e soprattutto leggera. Infatti, a differenza delle altre distribuzioni Linux forensi che si basano su Ubuntu, Debian, Arch Linux e similari, Tiny Core Forensic Edition si basa su Tiny Core Linux, che è una distribuzione Linux il cui Core ha una dimensione di appena 11-16 MB.

La leggerezza di Tiny Core Forensic Edition, rende tale distribuzione innovativa e attualmente unica nel panorama delle distribuzioni Linux forensi.

3.1. Definizione del problema

Il seguente lavoro di tesi si pone sia l'obiettivo di aggiornare la versione di Tiny Core alla base del sistema operativo, nonché il kernel Linux sia la realizzazione di un'installer, nonché una procedura di installazione, che permetta di installare Tiny Core Forensic Edition su PC fisici o macchine virtuali.

In primis si procederà dunque ad aggiornare Tiny Core all'ultima versione disponibile, ovvero la 14.0 (con kernel Linux 6.1.2-tinycore), aggiornando eventualmente anche le utility forensi in caso di incompatibilità con le nuove versioni del Core.

In seguito, si procederà alla realizzazione di un installer che fornisca una procedura guidata e automatizzata per installare il sistema operativo su PC fisici o su macchine virtuali. L'installer includerà necessariamente anche funzionalità dedicate alla formattazione del disco e creazione delle

partizioni, in modo tale da poter consentire agli utenti di personalizzare l'installazione del sistema operativo.

In tal modo Tiny Core Forensic Edition sarà sia eseguibile in modalità “Live”, sia installabile eventualmente su disco rigido.

3.2. Aggiornamento del Core

Come anticipato nel paragrafo precedente, il primo obiettivo di questo lavoro di tesi è aggiornare il Core alla base di Tiny Core Forensic Edition alla versione 14.0 (con kernel Linux 6.1.2-tinycore). Tale aggiornamento tuttavia non deve compromettere il normale funzionamento delle utility forensi incluse nella distribuzione, garantendo dunque la compatibilità con esse. Qualora un'applicazione non fosse compatibile con il Core aggiornato, si provvederà ad aggiornarla, e, se necessario, ricompilarla.

Infine, oltre all'aggiornamento del Core in sé, si provvederà ad aggiornare anche eventuali librerie nonché Python, che verrà aggiornato alla versione 3.9.

Tuttavia, per motivi di compatibilità, sul sistema sarà presente sia Python versione 3.6, sia Python versione 3.9.

Oltre agli aggiornamenti descritti sopra, anche il tool Volatility 3 è stato aggiornato all'ultima versione. Maggiori dettagli sull'aggiornamento di tale tool sono illustrati nel paragrafo seguente.

3.2.1. Aggiornamento di Volatility 3

Volatility 3 è un tool abbastanza utile in ambito forense, in quanto permette il recupero dei dati presenti nelle memorie portatili come le memorie RAM. Uno dei punti di forza di questo software è l'indipendenza dalle API native dei vari sistemi operativi. In tal modo esso è utilizzabile in maniera del tutto

indipendente dal sistema in esecuzione sul dispositivo sul quale si desidera effettuare il recupero dati.²¹

Il tool Volatility 3 fu incluso in Tiny Core Forensic Edition dalla studentessa Ruffo S. ed è presente sin dalla versione 1.0 del sistema, ma non è stato più aggiornato nelle release successive.

In questo paragrafo viene dunque descritta la procedura di aggiornamento di Volatility 3 all'interno di Tiny Core Forensic Edition.

Innanzitutto va precisato che Volatility 3 utilizza il linguaggio di programmazione Python e non viene distribuito come pacchetto TCZ ma è direttamente eseguibile una volta scaricato.

Per scaricare il software è necessaria però l'installazione del software Git (pacchetto git.tcz in Tiny Core), in modo da poter clonare in locale il repository Git del progetto presente su GitHub.

Per questioni di uniformità con le precedenti release di Tiny Core Forensic Edition, il repository è stato clonato all'interno della cartella home (cartella in cui era presente Volatility 3 nelle precedenti release): dunque, dopo essersi posizionati in `/home/tc`, si è proceduto ad effettuare il clone del repository tramite il seguente comando:

```
$ git clone  
https://github.com/volatilityfoundation/volatility3.git
```

Infine, per eseguire Volatility 3, è necessario recarsi nella cartella `/home/tc/volatility3` ed eseguire il comando:

```
$ python3 vol.py
```

²¹ <https://github.com/volatilityfoundation/volatility3>

3.3. Tiny Core Forensic Edition Installer

Al fine di estendere notevolmente le funzionalità e gli scenari d'uso di Tiny Core Forensic Edition, l'ulteriore obiettivo di questo lavoro di tesi è la realizzazione di un installer, che possa dunque permettere all'utente finale di installare il sistema operativo in maniera semplice e guidata. L'installer ovviamente sarà dotato di interfaccia grafica e tale interfaccia sarà progettata in maniera da essere semplice e intuitiva nei confronti dell'utente finale.

Allo stesso tempo, in base alla volontà e all'esperienza dell'utente, sarà possibile avviare una procedura di installazione quasi totalmente automatizzata, e una procedura di installazione personalizzata. La prima gestirà autonomamente il partizionamento del disco, nonché la formattazione, la seconda permetterà all'utente di scegliere il disco di destinazione, di partizionare il disco, e, infine, di scegliere la partizione sul disco in cui installare il sistema.

L'interfaccia grafica dell'installer prende spunto dagli installer utilizzati per distribuire software in ambiente Windows: tali installer difatti presentano un'interfaccia grafica piuttosto semplice e funzionale.

La procedura di installazione sarà presente in una voce a parte nel menu di avvio della ISO: in tal modo viene mantenuta una separazione netta con la modalità Live del sistema operativo.

All'avvio della modalità installazione verrà esclusivamente mostrato lo sfondo di Tiny Core Forensic Edition insieme alla schermata di benvenuto dell'installer.

3.3.1. Requisiti

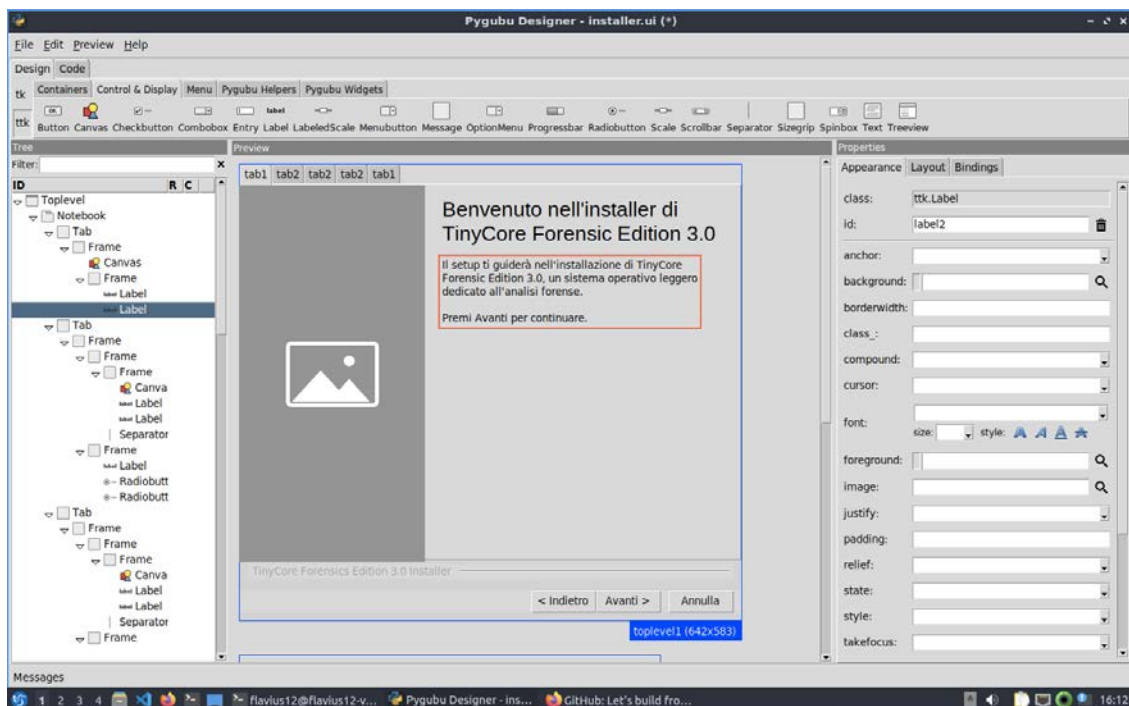
Nella realizzazione dell'installer, dovranno essere rispettati i seguenti requisiti:

- Modalità installazione separata dalla modalità Live
- La modalità di installazione deve essere dotata di interfaccia grafica
- L'interfaccia grafica deve essere semplice e intuitiva
- Installazione guidata e automatizzata per utenti meno esperti
- Installazione personalizzata per utenti avanzati

3.3.2. Interfaccia grafica

L'installer, come anticipato, segue la tipica struttura degli installer utilizzati per distribuire software in ambiente Windows.

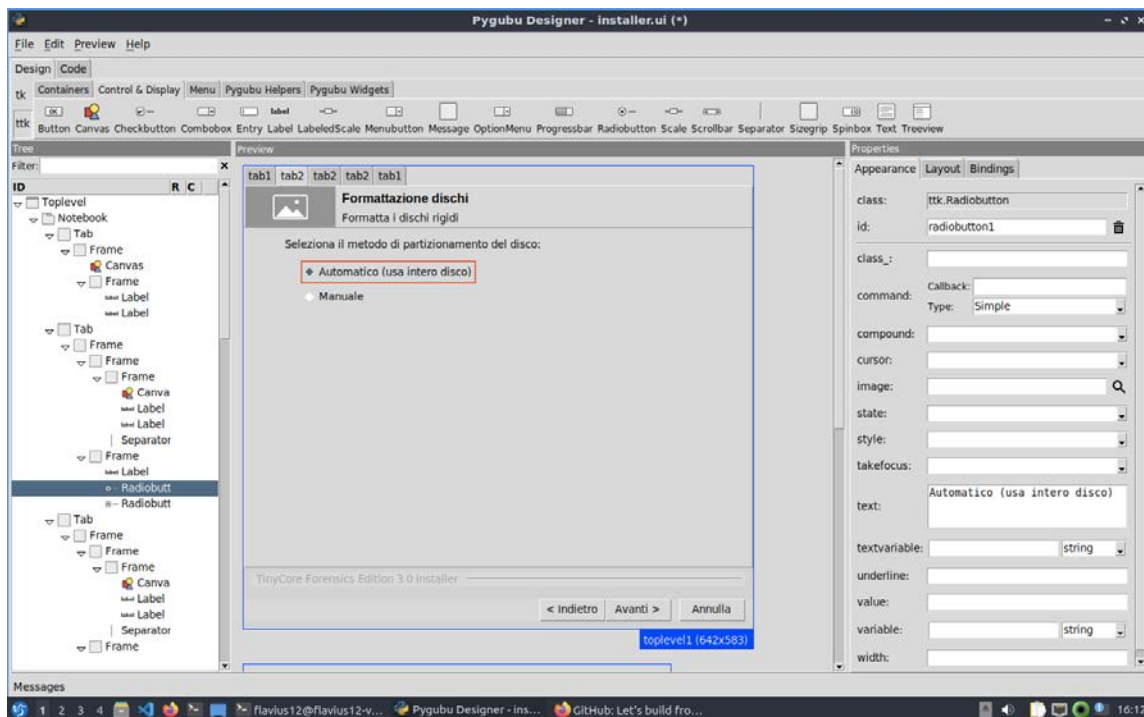
Il primo frame che verrà mostrato all'avvio dell'installer è la schermata di benvenuto:



Tale schermata introduce l'utente nella procedura di installazione, mostrando la versione specifica di Tiny Core Forensic Edition che si andrà ad installare e sintetizzando i passi che egli dovrà seguire.

La schermata di benvenuto è seguita dalla schermata di selezione della modalità di partizionamento. Sono disponibili due modalità di partizionamento:

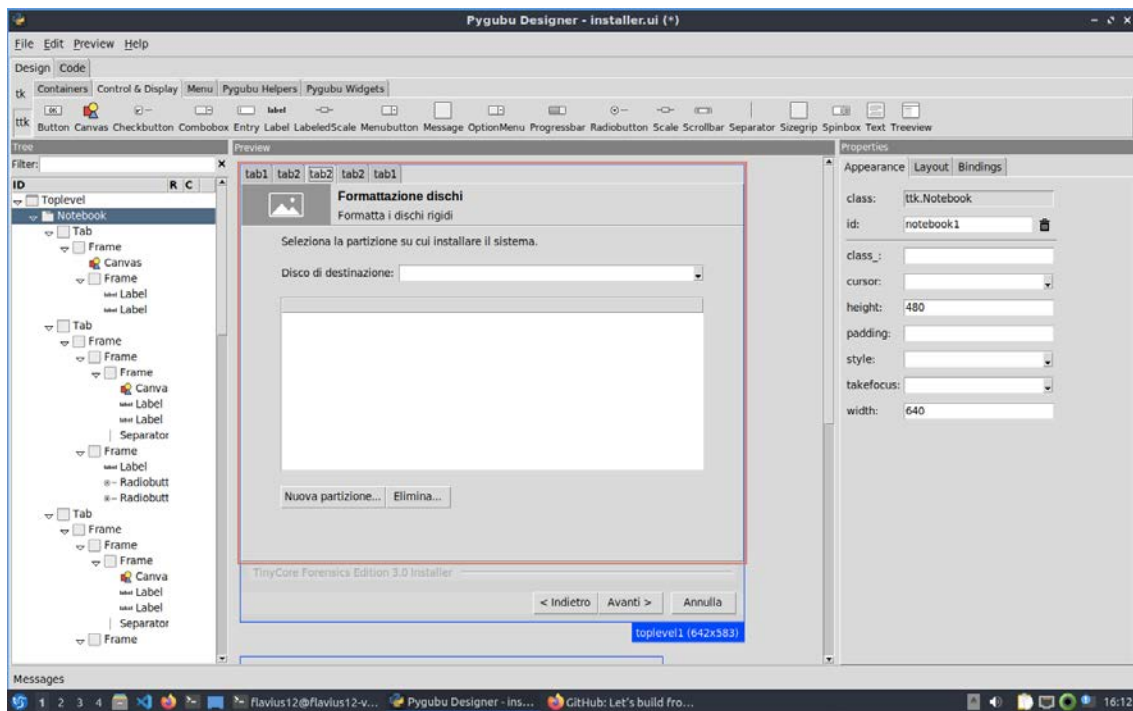
- *Modalità automatica:* dedicata agli utenti novizi, questa modalità installa il sistema operativo e i relativi dati, dopo aver selezionato e partizionato automaticamente il disco di destinazione;
- *Modalità manuale:* dedicata ad utenti più esperti, questa modalità permette di selezionare il disco in cui installare il sistema, nonché partizionare tale disco come si desidera. Tale modalità è utile, inoltre, qualora si voglia installare Tiny Core Forensic Edition come sistema operativo secondario sull'host (es. in modalità dual-boot con Windows o Linux).



In base alla modalità scelta, verrà mostrata la schermata di partizionamento dei dischi o si procederà direttamente all'installazione.

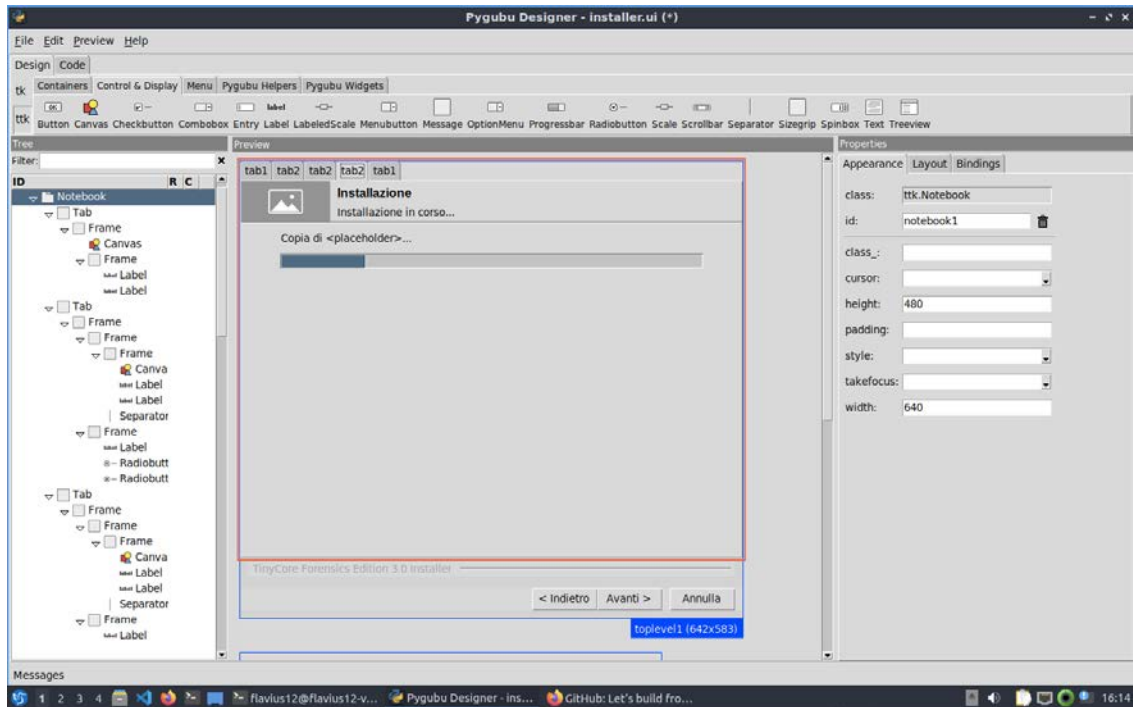
La schermata di partizionamento dischi fornisce le funzioni base per la creazione/cancellazione di partizioni, nonché formattazione di esse.

In particolare è possibile creare sia partizioni primarie che logiche e selezionare la partizione in cui installare, nella fase successiva, il sistema.

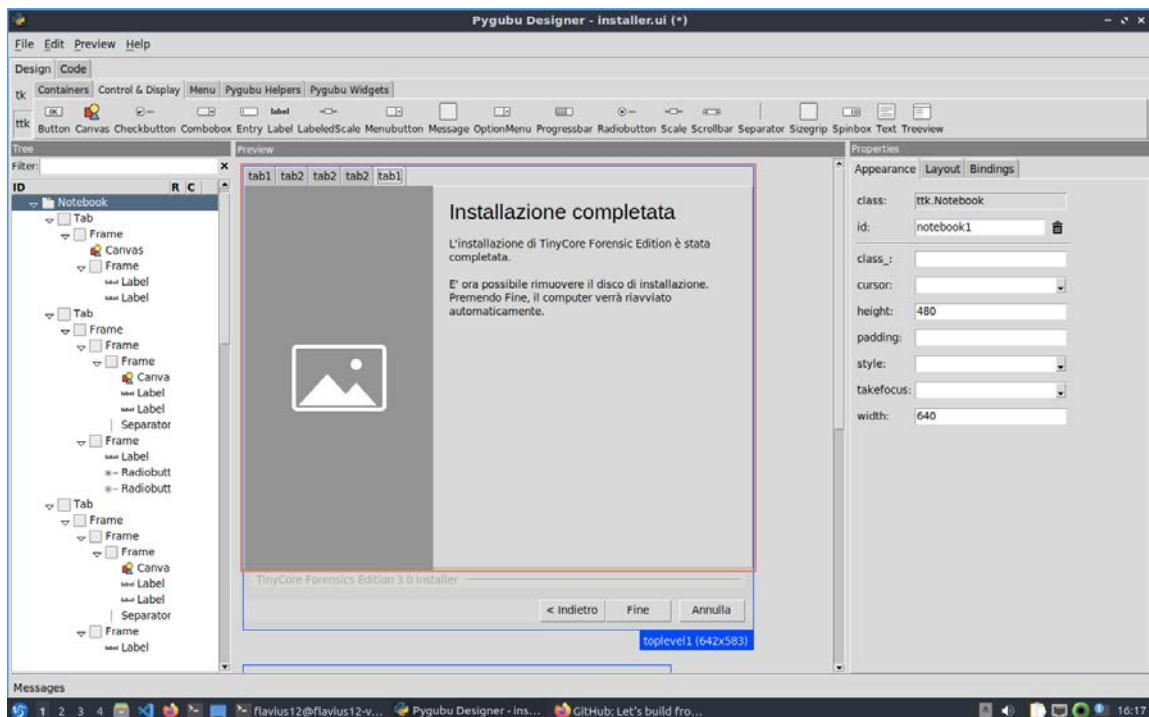


Prima di procedere all'installazione, per entrambe le modalità viene chiesta conferma all'utente di proseguire con il partizionamento del disco, nonché formattazione della partizione scelta. Tale avviso è mostrato in quanto le operazioni di partizionamento e formattazione del disco comportano la cancellazione di eventuali dati presenti su di esso.

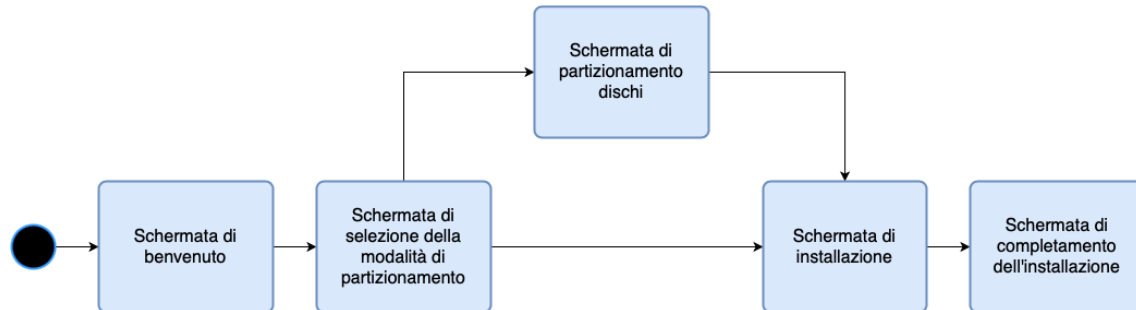
A seguito della formattazione, viene mostrata la schermata di installazione. Tale schermata mostra lo stato di installazione dei componenti del sistema e del bootloader tramite una progress bar. Al di sopra della progress bar è presente una label che mostra l'operazione corrente.



Al termine della copia dei componenti del sistema operativo, viene mostrata la schermata di completamento dell'installazione, nella quale viene indicato all'utente di rimuovere il disco di installazione. Premendo sul pulsante "Fine", si procederà al riavvio della macchina, decretando il completamento della fase di installazione del sistema.



Riassumendo, la sequenza dei frame è la seguente:



4. Implementazione

Il software di installazione è stato implementato in Python, un linguaggio di programmazione molto popolare grazie alla sua semplicità, modularità e portabilità.

La scelta di Python è dettata dall'ulteriore vantaggio dovuto alla presenza dell'interprete all'interno di Tiny Core: in questo modo è possibile eseguire gli script Python in maniera piuttosto semplice, senza dover compilare software aggiuntivo.

Tuttavia, vista la complessità del software che si intende realizzare, è necessario l'utilizzo di alcune librerie di supporto per implementare tutte le funzionalità richieste.

4.1. Prerequisiti

Per il corretto funzionamento del software di installazione, è richiesto Python versione 3.9 o successiva.

Per quanto riguarda l'implementazione dell'interfaccia grafica e delle funzionalità avanzate, sono state utilizzate le seguenti librerie:

- **Tkinter:** Tkinter (Tk-interface) è una libreria Python utilizzata per lo sviluppo di software che necessitano di un'interfaccia grafica. Essa è l'interfaccia Python standard per il toolkit GUI Tcl/Tk. Sia Tk che Tkinter sono disponibili su Windows, Linux e macOS;²²
- **Pillow:** Pillow è un fork aggiornato di Python Imaging Library (PIL), una libreria aggiuntiva gratuita e open source per il linguaggio di programmazione Python che aggiunge il supporto per l'apertura, la manipolazione e il salvataggio di numerosi formati di file immagine. È disponibile per Windows, Linux e macOS;²³
- **pyParted:** pyparted è un insieme di collegamenti Python nativi per libparted, libreria alla base del progetto GNU parted. Con pyparted è possibile scrivere applicazioni che interagiscono e gestiscono le partizioni del disco nonché i filesystem;²⁴
- **GNU Parted:** GNU Parted è un software in grado di manipolare le tabelle delle partizioni del disco. Ciò è utile per creare e cancellare partizioni, copiare i dati sui dischi rigidi, nonché riorganizzare l'utilizzo del disco. Il pacchetto contiene una libreria, libparted, ed un frontend a riga di comando, parted, che può essere utilizzato anche negli script.²⁵

Un'ulteriore requisito è la presenza del pacchetto grub2-multi.tcz, disponibile nel repository ufficiale di Tiny Core. Tale pacchetto fornisce le utility necessarie all'installazione del bootloader, come il tool grub-install.

²² <https://docs.python.org/3/library/tkinter.html>

²³ https://en.wikipedia.org/wiki/Python_Imaging_Library

²⁴ <https://github.com/dcantrell/pyparted>

²⁵ <https://www.gnu.org/software/parted/>

4.1.1. Compilazione di termcap

Termcap è una libreria e un database utilizzato su computer Unix-like. Essa consente ai programmi di utilizzare i terminali dei computer in modo indipendente dal dispositivo, il che semplifica notevolmente il processo di scrittura di applicazioni a linea di comando.²⁶

Tale libreria è utilizzata da GNU Parted, pertanto è necessario installarla per compilare correttamente tale software: in assenza di termcap infatti, durante la compilazione di GNU Parted, verrà mostrato il seguente errore:

```
termcap could not be found
```

Per compilare termcap, si procede innanzitutto a scaricarlo il sorgente con wget:

```
$ wget ftp.gnu.org/gnu/termcap/termcap-1.3.1.tar.gz
```

Una volta scaricato l'archivio contenente il codice sorgente, si procede all'estrazione tramite il comando tar e ci si reca nella directory appena estratta:

```
$ tar xvf termcap-1.3.1.tar.gz  
$ cd termcap-1.3.1
```

Prima di procedere alla compilazione è necessario installare il compilatore GNU, disponibile come pacchetto denominato compiletc.tcz.

Una volta installato il pacchetto compiletc, è possibile procedere con la compilazione della libreria e creazione del relativo tcz.

²⁶ <https://en.wikipedia.org/wiki/Termcap>

Innanzitutto, si procede creando la cartella di lavoro temporanea:

```
$ mkdir /tmp/termcap
```

Tale cartella conterrà i file e le directory che verranno poi inseriti all'interno del pacchetto tcz.

Una volta creata la cartella di lavoro temporanea, si procede con la compilazione e installazione del software:

```
$ ./configure --prefix=/usr/local  
$ touch /tmp/mark  
$ make install DESTDIR=/tmp/termcap
```

Tra il comando `configure` e il comando `make install`, viene eseguito il comando `touch`. Tale comando serve per distinguere i file che verranno installati a seguito della compilazione del software da tutto ciò che è già presente nel sistema.

Una volta terminata la compilazione e installazione del software, si procede con la creazione del pacchetto tcz.

Innanzitutto viene creato un file temporaneo (`/tmp/list`) contenente una lista dei file installati a seguito della compilazione:

```
$ find /usr/local -newer /tmp/mark -not -type d > /tmp/list
```

Tale lista viene utilizzata per creare un archivio temporaneo, che verrà successivamente estratto all'interno della cartella di lavoro temporanea `/tmp/termcap`:

```
$ tar -T /tmp/list -czvf termcap.tar.gz  
$ cd /tmp/termcap  
$ tar -xf /tmp/termcap.tar.gz  
$ cd /tmp
```

Una volta estratto il contenuto di tale archivio, si procede alla creazione del pacchetto tcz. Per creare file tcz, è necessario innanzitutto installare il tool mksquashfs (disponibile all'interno del pacchetto squashfs-tools.tcz).

Una volta installata l'utility, è possibile procedere alla creazione del pacchetto tcz tramite il seguente comando:

```
$ mksquashfs termcap/ termcap.tcz
```

4.1.2. Compilazione di GNU Parted

Per il corretto funzionamento della libreria py parted, è necessario che sia correttamente compilata ed installata la libreria libparted (parte del pacchetto GNU Parted). Tale libreria tuttavia non è presente in Tiny Core, né è scaricabile dall'Application Manager.

Per questo motivo è stato necessario scaricare il codice sorgente di GNU Parted per poi compilarlo su Tiny Core.

Per prima cosa si è proceduto ad installare il software Git (pacchetto git.tcz) su Tiny Core, in modo da poter effettuare la clonazione del codice sorgente di GNU Parted dal repository ufficiale (<https://git.savannah.gnu.org/git/parted.git>).

Pertanto, una volta spostati nella home directory di Tiny Core, è stato eseguito il seguente comando:

```
$ git clone https://git.savannah.gnu.org/git/parted.git
```

Una volta clonato il codice sorgente di GNU Parted, per procedere alla compilazione è necessario innanzitutto installare i seguenti tool oltre a quelli descritti nel paragrafo precedente:

- Autoconf
- Automake

Entrambi i tool sono disponibili come pacchetti TCZ nel repository di Tiny Core rispettivamente come `autoconf.tcz` e `automake.tcz`.

Un'ulteriore utility richiesta è `wget`: sebbene ci sia già una versione inclusa in Tiny Core come parte del pacchetto `BusyBox`, essa non include tutte le funzionalità necessarie. Pertanto, senza installare il pacchetto `wget.tcz`, lo script bootstrap restituirà l'errore:

```
wget: unrecognized option '--mirror'
```

durante lo scaricamento del sorgente di `Gnulib`.

`Gnulib`, inoltre, richiede la presenza di una delle due librerie tra `glibtoolize` e `libtoolize`, pertanto è necessario installare il pacchetto `libtool.tcz`. In assenza del pacchetto `libtool.tcz`, lo script bootstrap restituirà l'errore:

```
one of these is required: glibtoolize libtoolize
```

Una volta installato il compilatore e le utility necessarie, è possibile eseguire il seguente comando:

```
$ ./bootstrap
```

Tale comando mostra le dipendenze mancanti per poter compilare il software.

Dall'esecuzione del comando è emersa la necessità di installare le seguenti dipendenze:

- `bc.tcz`
- `gperf.tcz`

- texinfo.tcz
- rsync.tcz

Oltre alle dipendenze mostrate dallo script bootstrap, è necessario installare manualmente altre tre dipendenze:

- libdevmapper, incorporata nel pacchetto liblvm2.tcz;
- headers di libdevmapper, incorporati nel pacchetto liblvm2-dev.tcz;
- readline, incorporata nel pacchetto readline.tcz;
- headers di readline, incorporati nel pacchetto readline-dev.tcz;
- termcap.

Una volta installate le dipendenze sopracitate, lo script bootstrap procederà con lo scaricamento del codice sorgente di un'ulteriore dipendenza chiamata GnuLib, nonché la compilazione di essa.

Al termine di esecuzione dello script bootstrap, si procede con la creazione della cartella di lavoro temporanea per il pacchetto tcz, compilazione e installazione del software:

```
$ mkdir /tmp/parted
$ ./configure --prefix=/usr/local --disable-debug
$ touch /tmp/mark
$ make install DESTDIR=/tmp/parted
```

Al termine della compilazione e installazione di parted, si procede con la generazione del pacchetto tcz:

```
$ find /usr/local -newer /tmp/mark -not -type d > /tmp/list
$ tar -T /tmp/list -czvf parted.tar.gz
$ cd /tmp/parted
$ tar -xf /tmp/parted.tar.gz
$ cd /tmp
$ mksquashfs parted/ parted.tcz
```

4.1.3. Python3.9 pyparted

Pyparted è una libreria Python che consente di utilizzare le API di GNU Parted in Python. Il codice sorgente è disponibile su GitHub, al seguente URL: <https://github.com/dcantrell/pyparted>.

Anche pyparted non è presente nel repository ufficiale di Tiny Core, pertanto è necessario creare manualmente il pacchetto tcz al fine di integrare la libreria all'interno del sistema.

Prima di procedere con la compilazione e installazione della libreria, è necessario verificare la presenza dei seguenti pacchetti:

- compiletc.tcz
- pkg-config.tcz
- python3.9-dev.tcz
- python3.9-setuptools.tcz
- python3.9.tcz
- parted.tcz (la cui generazione è descritta nel paragrafo precedente)

Per semplificare l'installazione di pyparted, è stato utilizzato il Python Package Manager pip. Tuttavia, poiché pip non è presente nel repository di Tiny Core, è necessario scaricarlo e installarlo manualmente. Per fare ciò è innanzitutto necessario installare il pacchetto expat2.tcz, dopodiché è possibile scaricare e installare pip tramite i seguenti comandi:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ python3 get-pip.py
```

A questo punto è possibile procedere con l'installazione del modulo pyparted. Tuttavia, poiché è necessario generare il pacchetto tcz di tale

modulo, si adotta una procedura simile a quella descritta nei paragrafi precedenti.

Per prima cosa viene generata la cartella di lavoro temporanea:

```
$ mkdir /tmp/python3.9-pyparted
```

Ora è possibile installare il modulo con pip, tale modulo sarà installato in un terminale root:

```
# pip3 install pyparted
```

Una volta completata l'installazione, si procede con la generazione del pacchetto python3.9-pyparted.tcz:

```
$ find /usr/local/lib/python3.9/site-packages/parted  
/usr/local/lib/python3.9/site-packages/pyparted-3.12.0.dist-  
info/ /usr/local/lib/python3.9/site-packages/_ped.cpython-39-  
i386-linux-gnu.so > /tmp/list  
$ tar -T /tmp/list -czvf /tmp/python3.9-pyparted.tar.gz  
$ cd /tmp/python3.9-pyparted  
$ tar -xf /tmp/python3.9-pyparted.tar.gz  
$ cd /tmp  
$ mksquashfs python3.9-pyparted/ python3.9-pyparted.tcz
```

4.1.4. Python3.9 tk

Il modulo Python Tkinter (Tk-interface) è utilizzato per la generazione dell'interfaccia grafica dell'installer. Essa è l'interfaccia Python standard per il toolkit GUI Tcl/Tk. Per installare Python 3.9 Tk è necessario che siano presenti i seguenti pacchetti:

- python3.9.tcz
- tk8.6.tcz

Una volta installati i requisiti sopracitati, è possibile procedere con l'installazione del modulo Python tkinter e creazione del relativo pacchetto tcz:

```
$ mkdir /tmp/python3.9-tk
$ touch /tmp/mark
$ pip3 install tk
$ find /usr/local/lib/python3.9/site-packages/ -newer
/tmp/mark -not
-type d > /tmp/list
$ tar -T /tmp/list -czvf /tmp/python3.9-tk.tar.gz
$ cd /tmp/python3.9-tk
$ tar -xf /tmp/python3.9-tk.tar.gz
$ cd /tmp
$ mksquashfs python3.9-tk/ python3.9-tk.tcz
```

4.1.5. Python3.9 pillow

La libreria Pillow fornisce le funzionalità necessarie per l'apertura, la manipolazione e il salvataggio di numerosi formati di file immagine. Tale libreria viene usata all'interno dell'installer per mostrare le immagini all'interno dell'interfaccia grafica.

La libreria Pillow non è disponibile nel repository ufficiale di Tiny Core, pertanto il pacchetto tcz deve essere creato manualmente.

Come prerequisiti per installare Pillow è necessario che siano presenti nel sistema i seguenti pacchetti, scaricabili dal repository ufficiale:

- python3.9-dev.tcz
- libjpeg-turbo-dev.tcz
- compiletc.tcz

Una volta installati i requisiti sopracitati, è possibile procedere con l'installazione del modulo Pillow e creazione del relativo pacchetto tcz:

```
$ mkdir /tmp/python3.9-pillow
$ touch /tmp/mark
$ pip3 install pillow
$ find /usr/local/lib/python3.9/site-packages/ -newer
```

```
/tmp/mark -not -type d > /tmp/list  
$ tar -T /tmp/list -czvf /tmp/python3.9-pillow.tar.gz  
$ cd /tmp/python3.9-pillow  
$ tar -xf /tmp/python3.9-pillow.tar.gz  
$ cd /tmp  
$ mksquashfs python3.9-pillow/ python3.9-pillow.tcz
```

È bene sottolineare che a runtime pillow richiede la presenza della libreria libjpeg, dunque è necessario installare il pacchetto libjpeg-turbo.tcz.

4.2. Realizzazione dell'interfaccia grafica

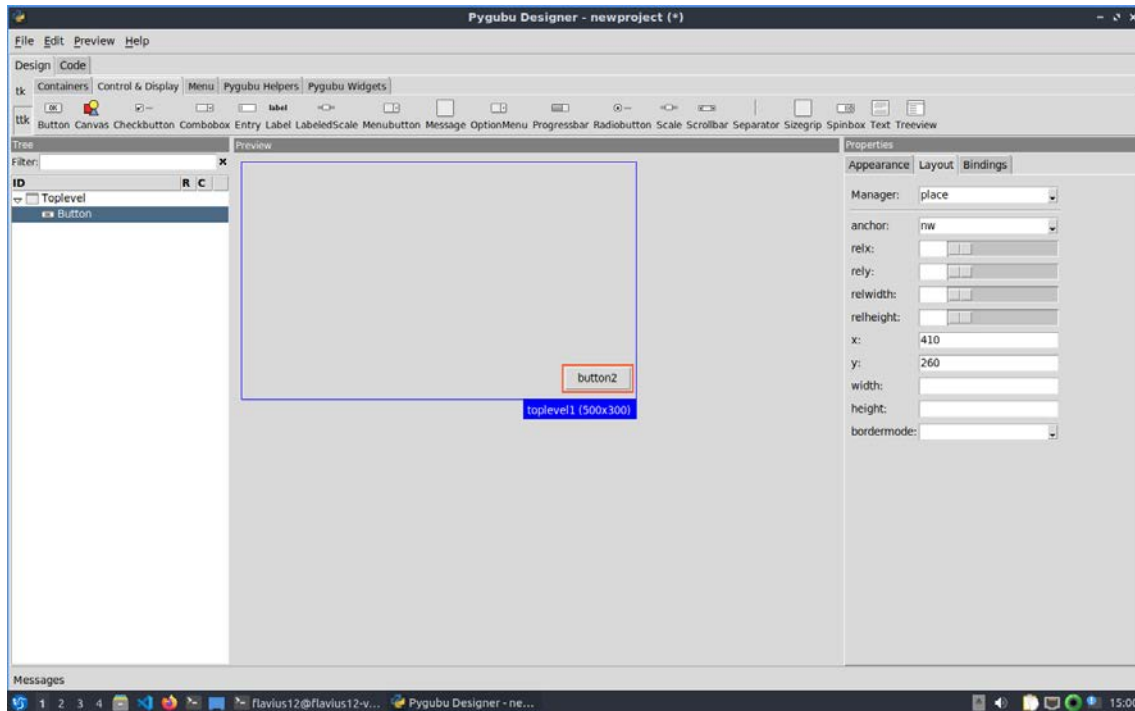
Per la realizzazione dell'interfaccia grafica, è stato utilizzato un programma Python chiamato Pygubu Designer.

Pygubu Designer è tool RAD (Rapid Application Development) per la realizzazione di interfacce grafiche cross-platform, utilizzando la libreria Tkinter.

Il software, basato su Python, è completamente gratuito e open source ed è disponibile al seguente repository GitHub.²⁷

Pygubu Designer supporta gran parte dei widget grafici inclusi in Tkinter, permettendone inoltre la personalizzazione dei parametri tramite la sidebar presente a destra dell'interfaccia di Pygubu. Le interfacce grafiche che Pygubu permette di realizzare possono essere salvate in formato XML oppure possono essere direttamente esportate in codice Python tramite l'applicazione.

²⁷ <https://github.com/alejandroautalan/pygubu-designer>



4.3. Analisi del codice

Il codice è suddiviso in diverse classi: ogni classe implementa un frame dell'installer, nonché le relative funzionalità.

In particolare, sono state definite le seguenti classi principali:

- **InstallerApp (file InstallerApp.py):** classe che implementa il core dell'installer. Essa inizializza la finestra principale dell'installer e coordina la navigazione tra i frame;
- **WelcomePage (file WelcomePage.py):** classe che implementa il frame di benvenuto;
- **DiskFormatPage (file DiskFormatPage.py):** classe che implementa il frame della selezione della modalità di partizionamento del disco;
- **CustomDiskFormatPage (file DiskFormatPage.py):** classe che implementa il frame di partizionamento personalizzato dei dischi;
- **InstallPage (file InstallPage.py):** classe che implementa il frame dello stato di installazione del sistema;

- **FinishPage (file FinishPage.py):** classe che implementa il frame del completamento dell'installazione.
- **RebootInfoApp (file RebootInfoApp.py):** classe che mostra una messagebox preposta ad informare l'utente sul termine dell'installazione e sul successivo riavvio del computer. Tale classe viene avviata in un'istanza Python separata da InstallerApp.

4.3.1. Gestione delle partizioni

L'installer permette all'utente di gestire sia il disco che la partizione di destinazione per l'installazione del sistema. In particolare è possibile delegare la gestione del partizionamento del disco all'installer stesso selezionando il partizionamento automatico o gestire manualmente il partizionamento del disco selezionando il partizionamento manuale.

Nel primo caso, l'installer seleziona il disco primario e crea un'unica partizione primaria della dimensione dell'intero disco. Tale partizione viene inoltre formattata con filesystem ext3 e verrà scelta come destinazione del sistema operativo.

Selezionando l'opzione partizionamento manuale invece è possibile scegliere manualmente sia il disco di destinazione che la partizione di destinazione. Le partizioni possono dunque essere gestite manualmente, in particolare è possibile sia creare partizioni primarie o logiche, sia cancellare quelle esistenti.

Una volta confermata la partizione di destinazione, l'installer provvederà alla formattazione con filesystem ext3.

Le informazioni sulle partizioni, nonché la loro gestione sono ottenute tramite le API messe a disposizione dal modulo Python `pyparted`. Tale modulo è un wrapper Python della libreria GNU Parted.

Per la creazione delle partizioni sul disco viene utilizzata la funzione `addPartition` di `pyparted`. Tale funzione prende in input i parametri `partition` e `constraint`: il primo parametro specifica la struttura della partizione che si intende creare (geometria della partizione, tipo di partizione e eventuali flag)., il secondo parametro specifica i vincoli di allineamento dei settori sul disco.

Una volta effettuate le operazioni sul disco o sulle partizioni, è necessario invocare la funzione `commit` per applicare le modifiche.

4.3.2. Copia dei file

Una volta formattata la partizione di destinazione, l'installer copia i file dall'ISO alla partizione di destinazione.

In particolare vengono innanzitutto copiati i due file fondamentali del sistema `vmlinux` e `core.gz`: il primo è il file presente direttamente in Tiny Core, il secondo è la versione del `core.gz` personalizzata con il write blocker e ulteriori modifiche.

Dopo la copia dei componenti fondamentali del sistema, l'installer provvede a copiare le estensioni `tcz`. Le estensioni `tcz` sono quelle presenti nell'istanza di Tiny Core Forensic Edition in cui è in esecuzione l'installer stesso e non vengono caricate in memoria RAM.

Terminata la copia delle estensioni `tcz`, vengono copiate le estensioni `ondemand` ovvero `BlockDevGUI` e `SharePoint Downloader` e il tool `Volatility 3`, il quale non si presenta sotto forma di estensione, ma è una semplice directory che viene poi copiata nella home directory del sistema.

Completata la copia dei vari software, si procede con la copia delle personalizzazioni del sistema come il wallpaper di Tiny Core Forensic Edition ed alcune personalizzazioni grafiche del bootloader.

4.3.3. Installazione del bootloader

La copia dei file è seguita dall'installazione del bootloader GRUB, indispensabile per l'avvio del sistema installato.

Per installare il bootloader, viene eseguito il comando `grub-install`. Tale comando è parte del pacchetto `grub2-multi.tcz`.

In particolare, il comando `grub-install` viene eseguito come segue:

```
$ grub-install --boot-directory=<directory_di_boot>  
                  <disco_di_destinazione>
```

L'opzione `boot-directory` specifica in quale cartella il bootloader GRUB deve caricare i file di configurazione di GRUB. L'opzione descritta sopra è infine seguita dal parametro `<disco_di_destinazione>` che specifica il disco in cui verrà installato il bootloader (es. `/dev/sda1`).

Una volta installato il bootloader GRUB, è necessario configurarlo e personalizzarlo in base alle specifiche del sistema di destinazione. Le informazioni di configurazione del bootloader GRUB sono caricate dal file `grub.config` presente all'interno della cartella `/boot/grub`.

L'installer, dunque, crea automaticamente il file di configurazione GRUB (`grub.config`) e lo copia nella directory `/boot/grub` del disco di destinazione.

Il frammento di codice deputato alla creazione del file `grub.config` in base ai parametri specifici dell'hardware è il seguente (InstallPage.py, 227-250):

```
deviceUUID = getDeviceUUID("/dev/{}".format(self._params[1]))  
grubConfigFile =  
open("{} /boot/grub/grub.cfg".format(self._params[0]), "w")  
grubConfigFile.write("set timeout=10\n")  
grubConfigFile.write("set timeout_style=menu\n")  
grubConfigFile.write("insmod ext3\n")  
grubConfigFile.write("insmod all_video\n")  
grubConfigFile.write("insmod efi_gop\n")  
grubConfigFile.write("insmod efi_uga\n")
```

```
grubConfigFile.write("insmod ieee1275_fb\n")
grubConfigFile.write("insmod vbe\n")
grubConfigFile.write("insmod vga\n")
grubConfigFile.write("insmod video_bochs\n")
grubConfigFile.write("insmod video_cirrus\n")
grubConfigFile.write("insmod gfxterm\n")
grubConfigFile.write("insmod png\n")
grubConfigFile.write("terminal_output gfxterm\n")
grubConfigFile.write("search --no-floppy --fs-uuid --set=root\n".format(deviceUUID))
grubConfigFile.write("loadfont\n/boot/grub/fonts/unicode.pf2\n")
grubConfigFile.write("background_image /boot/grub/TCF.png\n")
grubConfigFile.write("menuentry \"TinyCore Forensics Edition\"\n{\n")
grubConfigFile.write("\tlinux /boot/vmlinuz quiet opt=UUID={}\n".format(deviceUUID))
grubConfigFile.write("\thome=UUID={}\n".format(deviceUUID))
grubConfigFile.write("\ttce=UUID={}\n".format(deviceUUID))
grubConfigFile.write("\tinitrd /boot/core.gz\n")
grubConfigFile.write("}\n")
grubConfigFile.close()
```

Per ottenere l'UUID del disco, è stata definita la funzione `getDeviceUUID`.

```
def getDeviceUUID(device):
    response = Popen(["blkid", "-s", "UUID", "-o", "value",
                      device], stdout=PIPE).communicate()
    return response[0].decode("utf-8").replace("\n", "")
```

Tale funzione esegue in un thread seaparato il comando `blkid` di Linux e ne interpreta l'output.

`Blkid` è un comando che fa parte dei sistemi Linux e permette di ottenere informazioni (es. filesystem, tipo di contenuto del device...), attributi e metadati dei block device.

In particolare il comando `blkid` viene chiamato come segue:

```
$ blkid -s UUID -o value <disco>
```

L'opzione `-s` specifica quale attributo del block device mostrare. Nel caso specifico dell'installer, l'opzione `-s` verrà invocata con il parametro

“UUID”: in tal modo il comando restituirà l’identificatore univoco del disco in cui verrà installato il sistema.

L’opzione `-o` specifica il formato di output. Nel contesto corrente, l’opzione `-o` viene invocata con il parametro “value”, in quanto è necessario ottenere esclusivamente il valore corrispondente all’attributo specificato tramite l’opzione `-s`.

Infine viene specificato il disco da cui ottenere le informazioni richieste.

Un esempio di file di configurazione GRUB generato è mostrato di seguito:

```
set timeout=10
set timeout_style=menu
insmod ext3
insmod all_video
insmod efi_gop
insmod efi_uga
insmod ieee1275_fb
insmod vbe
insmod vga
insmod video_bochs
insmod video_cirrus
insmod gfxterm
insmod png
terminal_output gfxterm
search --no-floppy --fs-uuid --set root 14608243-3d88-403f-
92de-0332ff90f49b
loadfont /boot/grub/fonts/unicode.pf2
background_image /boot/grub/TCF.png
menuentry "TinyCore Forensics Edition"{
    linux /boot/vmlinuz quiet opt=UUID=14608243-3d88-403f-
92de-0332ff90f49b home=UUID=14608243-3d88-403f-92de-
0332ff90f49b tce=UUID=14608243-3d88-403f-92de-0332ff90f49b
    initrd /boot/core.gz
}
```

La proprietà `timeout` definisce un timeout di 10 secondi prima dell’avvio del sistema. Con la proprietà `timeout_style` si configura il bootloader in modo da mostrare un menu di avvio grafico prima di procedere con il caricamento del sistema. Le direttive `insmod` permettono di abilitare moduli specifici di GRUB. In particolare, sono stati abilitati i moduli per

implementare il supporto del filesystem ext3 e per implementare il menu grafico e lo sfondo del bootloader.

Tramite la direttiva `search`, GRUB cerca il disco di avvio direttamente tramite UUID. In questo modo viene identificato univocamente il disco anche in caso di swap delle unità a livello hardware.

Infine la direttiva `menuentry` definisce l'elemento del menu "TinyCore Forensics Edition", a cui vengono assegnati i comandi specifici per il caricamento di Tiny Core.

4.3.4. Script di supporto dell'installer

A supporto dell'installer, è stato creato uno script denominato `start.sh`. Tale script si occupa di eseguire l'installer con i privilegi di root (in assenza di tali privilegi non è possibile né gestire le partizioni, né installare il bootloader) nonché di riavviare il sistema una volta che l'installazione è completa. Lo script si compone di poche linee di codice ed ha il seguente contenuto:

```
#!/bin/sh
sudo python3 /usr/local/tcfe-setup/InstallerApp.py
sudo python3 /usr/local/tcfe-setup/RebootInfoApp.py
sudo reboot
```

La prima riga serve a informare l'interprete `sh` che il file seguente è uno script `sh`.

La seconda riga esegue l'installer con i privilegi di root. Una volta terminata l'esecuzione dell'installer, lo script esegue il file Python `RebootInfoApp.py`: tale file, come detto in precedenza, mostra una message box che informa l'utente circa la terminazione dell'installazione e l'esecuzione del riavvio automatico.

Una volta che l'utente chiude la message box, lo script provvede a riavviare il dispositivo.

4.3.5. Startup script

Per rendere l'installer autoavviabile quando si carica il sistema operativo è necessario creare uno script di avvio da copiare nella cartella `/home/tc/.X.d/`

Gli script contenuti in tale cartella saranno avviati automaticamente appena viene terminato il caricamento del desktop manager di sistema.

Nel caso specifico dell'installer, è stato creato il file `"tcfe-setup"`. Tale file non contiene altro che il percorso del file di supporto dell'installer (`start.sh`), descritto nel paragrafo precedente.

In questo modo, all'avvio verrà caricato il file `/usr/local/tcfe-setup/start.sh`, che a sua volta eseguirà l'installer.

4.4. Packaging dell'installer

La creazione del pacchetto `tcz` dell'installer segue la medesima procedura, con l'unica eccezione per lo startup script `tcfe-setup`.

In particolare lo script `tcfe-setup` non viene incluso nel pacchetto `tcz`, ma viene incluso nel backup di Tiny Core generato tramite il tool di backup integrato nel sistema (si veda 5.2 – Personalizzazione del Core per la modalità installazione).

Il resto dei file invece vengono copiati in `/usr/local/tcfe-setup`. In tal modo i file verranno estratti nel percorso sopracitato al caricamento dell'estensione `tcz`.

Infine vengono eseguiti i seguenti comandi per la creazione dell'estensione `tcfe-setup.tcz`:

```
$ mkdir /tmp/tcfe-setup
$ find /usr/local/tcfe-setup/ -not -type d > /tmp/list
$ tar -T /tmp/list -czvf /tmp/tcfe-setup.tar.gz
$ cd /tmp/tcfe-setup
$ tar -xf /tmp/tcfe-setup.tar.gz
$ cd /tmp
$ mksquashfs tcfe-setup/ tcfe-setup.tcz
```

5. Creazione della ISO

In questo capitolo vengono descritte nel dettaglio le personalizzazioni effettuate sia sul Core per la modalità Live sia su quello per la modalità di installazione.

Infine viene illustrato il tool EzRemaster, nonché i dettagli relativi alla generazione della ISO finale.

5.1. Personalizzazione del Core per la modalità Live

Per quanto riguarda la modalità Live, non sono state aggiunte funzionalità rispetto alla versione precedente di Tiny Core Forensic Edition. Tuttavia si è provveduto ad aggiornare il Core nonché alcuni tool forensi come Volatility 3.

Poiché nelle versioni precedenti di Tiny Core Forensic Edition erano presenti delle personalizzazioni come il wallpaper e il file di configurazione del Write Blocker, si è provveduto a mantenere tali personalizzazioni anche in questa versione di Tiny Core.

Per fare ciò, è stata creata una macchina virtuale con un'unità ottica virtuale e un disco rigido virtuale, contenente il wallpaper, denominato `TCF.png`, e il file di configurazione del Write Blocker, denominato `forensic.rules`. Tale macchina virtuale verrà utilizzata anche in seguito per la personalizzazione del Core per la modalità installazione.

Una volta creata la macchina virtuale, è stata avviata la ISO dell'ultima versione di Tiny Core ed è stato montato il disco virtuale con il comando:

```
$ mount /dev/sda1
```

A questo punto il file `TCF.png` è stato copiato dall'unità `sda1` in `/opt/backgrounds`, mentre il file `forensic.rules` è stato copiato dall'unità `sda1` in `/etc/udev/rules.d/`.

Una volta copiati i file è stato impostato lo sfondo del desktop su `TCF.png` tramite le impostazioni di Tiny Core.

Oltre alle personalizzazioni, si è proceduto a clonare il repository git di Volatility 3 all'interno della home directory.²⁸

Infine si è proceduto a modificare il file `filetool.lst`, presente in `/opt`, aggiungendo i percorsi dei file sopra descritti: in questo modo, effettuando il backup di Tiny Core tramite il tool integrato, verranno inclusi anche tali file.

Terminata la modifica del file `filetool.lst`, è stato effettuato il backup nell'unità `sda1`: l'operazione di backup ha generato un file denominato `mydata.tgz` all'interno della root di `sda1`. Tale file conterrà i file `TCF.png`, `forensic.rules` e la directory di Volatility 3, oltre ai file di sistema.

Il file `mydata.tgz` è stato infine rinominato in `mydata-live.tgz`, in modo da distinguerlo da quello che verrà generato a seguito del backup del Core della modalità installazione.

5.2. Personalizzazione del Core per la modalità installazione

Seguendo quanto specificato nei requisiti, è richiesto che la modalità installazione sia separata dalla modalità Live. Questa scelta è dettata sia dalla necessità di non sovraccaricare la memoria da estensioni che sarebbero pressoché inutilizzate, sia dalla presenza del Write Blocker nella modalità Live, che potrebbe interferire con la copia dei file durante l'installazione del sistema.

²⁸ <https://github.com/volatilityfoundation/volatility3>

Ciò comporta la necessità di avere due istanze di Tiny Core differenti e indipendenti per la modalità Live e per la modalità installazione.

A differenza della modalità Live, la modalità installazione deve necessariamente contenere i file e le personalizzazioni che l'installer dovrà copiare sul disco di destinazione. Anche in questo caso è stata effettuata un'operazione simile a quella descritta nel paragrafo differente.

Utilizzando la macchina virtuale descritta prima, sono stati copiati i file di personalizzazione, i file di personalizzazione del bootloader e i file di sistema. Tali file sono stati copiati all'interno della cartella home, che costituisce la cartella di appoggio dei file che il software di installazione copierà sul disco di destinazione.

In particolare sono stati innanzitutto copiati i file `core.gz` e `vmlinuz` dalla ISO di Tiny Core: tali file costituiscono i pilastri del sistema operativo Tiny Core.

Come per la versione Live, sono stati copiati i file `TCF.png` e `forensic.rules` ed è stato clonato Volatility 3 dal repository GitHub all'interno della cartella home. Dalla versione live inoltre è stato copiato anche il file `onboot.lst`, file che contiene la lista di tutte le estensioni che devono essere caricate all'avvio del sistema.

Ulteriori file copiati sono quelli relativi alla personalizzazione e configurazione del bootloader GRUB, tra cui il file `ascii.pf2`.

Per garantire inoltre che il software di installazione si apra all'avvio del sistema, è stato copiato il file `tcfe-setup`, descritto nel capitolo precedente, nella cartella `/home/tc/.X.d`.

Per quanto riguarda le utility BlockDevGUI e Sharepoint, si è proceduto ad estrarre manualmente i pacchetti nella cartella home, in quanto, essendo

estensioni ondemand, presentano non pochi problemi con la gestione dei path in fase di caricamento.

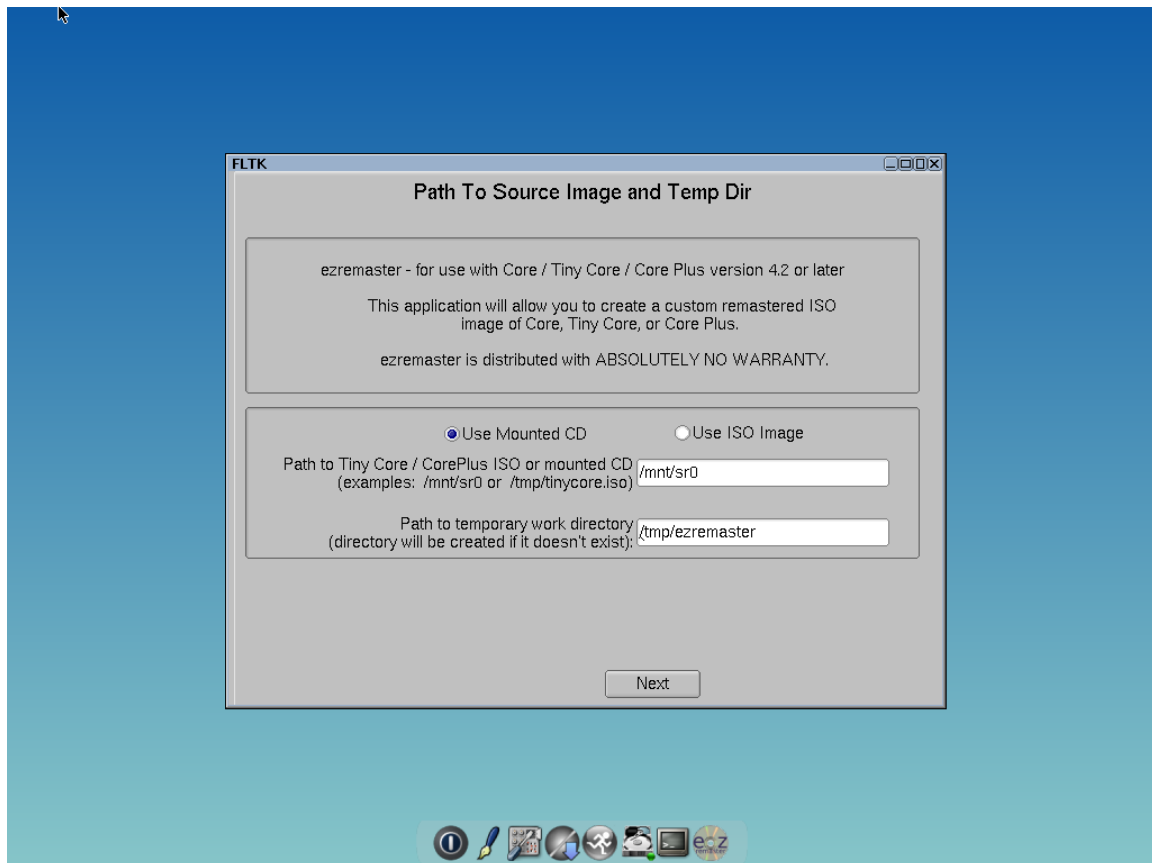
Terminata la copia dei file, si è proceduto ad impostare lo sfondo su `TCF.png` dalle impostazioni di sistema e si è modificato il file `filetool.lst`, aggiungendo i percorsi dei file sopra elencati.

A questo punto si è proceduto ad effettuare il backup nell'unità `sda1`: l'operazione di backup ha generato un file denominato `mydata.tgz` all'interno della root di `sda1`. Il file `mydata.tgz` è stato infine rinominato in `mydata-setup.tgz`.

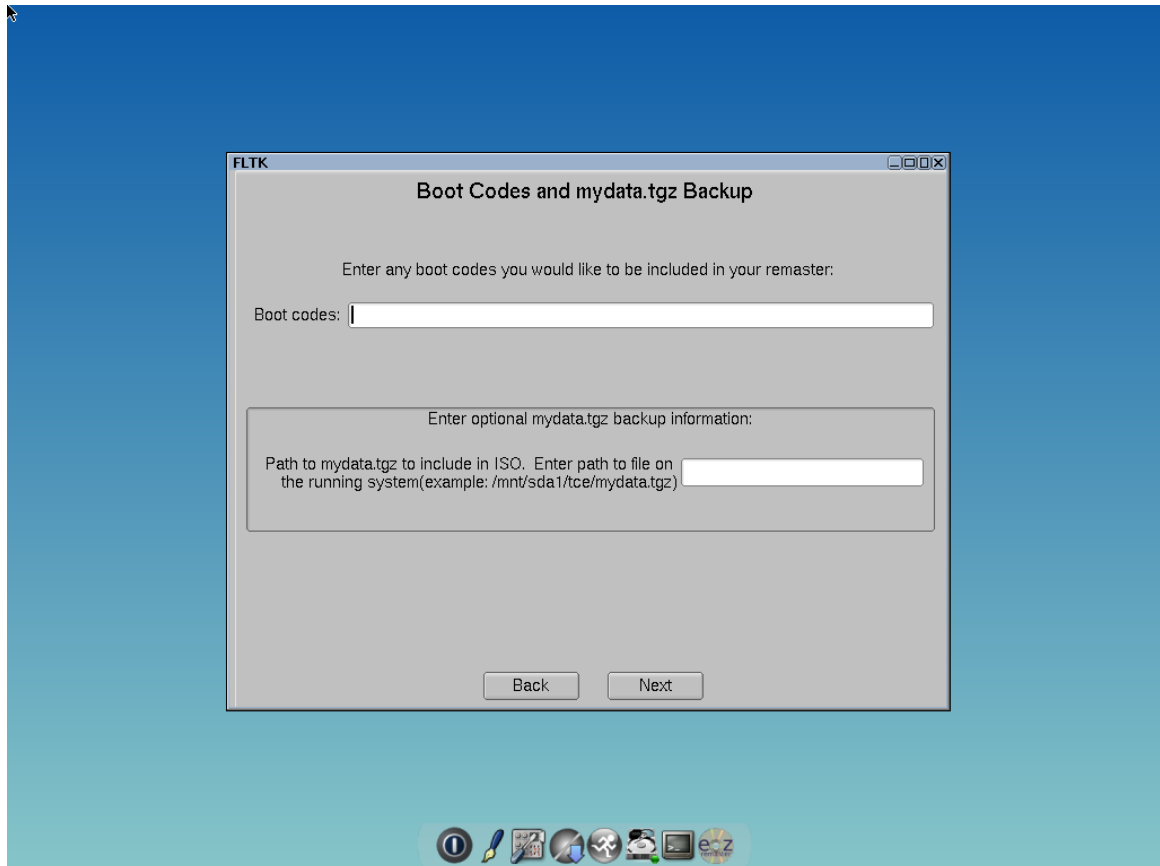
5.3. EzRemaster

Per effettuare il remastering, ovvero l'editing dell'immagine di sistema, è disponibile un tool abbastanza completo chiamato EzRemaster. Questo tool è utilizzabile sia da terminale che tramite interfaccia grafica ed è presente nel repository ufficiale di Tiny Core. Tuttavia, nonostante EzRemaster sia utilizzabile da terminale, gran parte delle funzionalità sono disponibili e accessibili tramite l'interfaccia grafica.

EzRemaster permette dunque di rimasterizzare la ISO di Tiny Core, partendo da quella già montata nel sistema o da un file ISO differente:

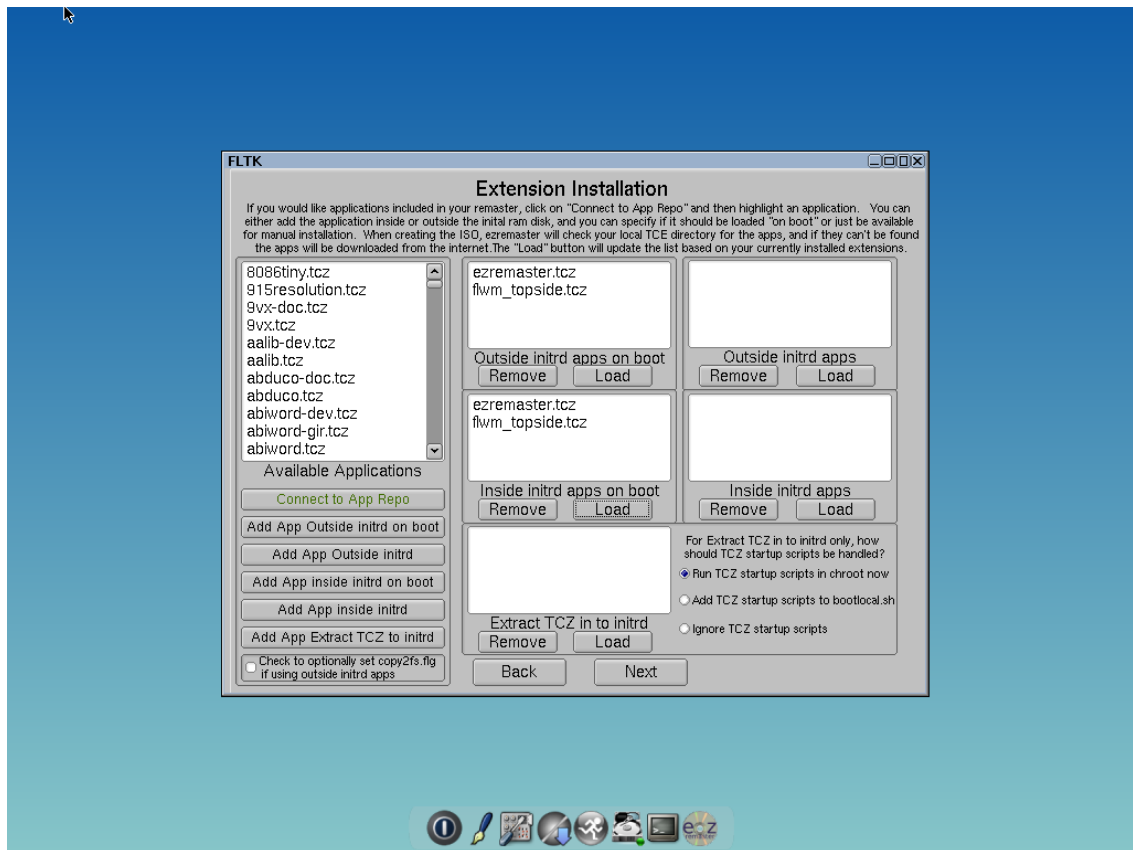


L'interfaccia grafica di EzRemaster permette inoltre di caricare un eventuale backup di sistema (file `mydata.tgz`):

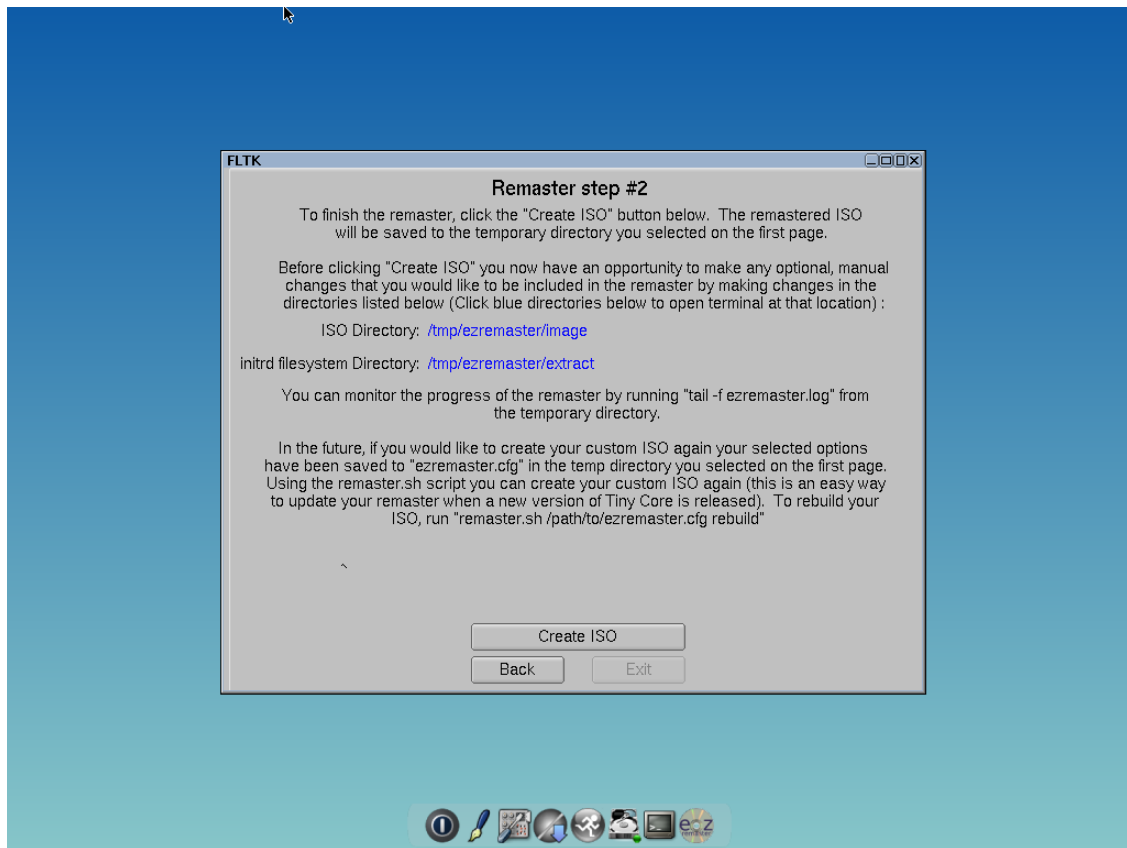


Successivamente, è possibile decidere quali estensioni copiare (e eventualmente caricare all'avvio) all'interno del sistema remasterizzato.

È bene tener presente che le estensioni devono essere in formato tcz per essere inserite.



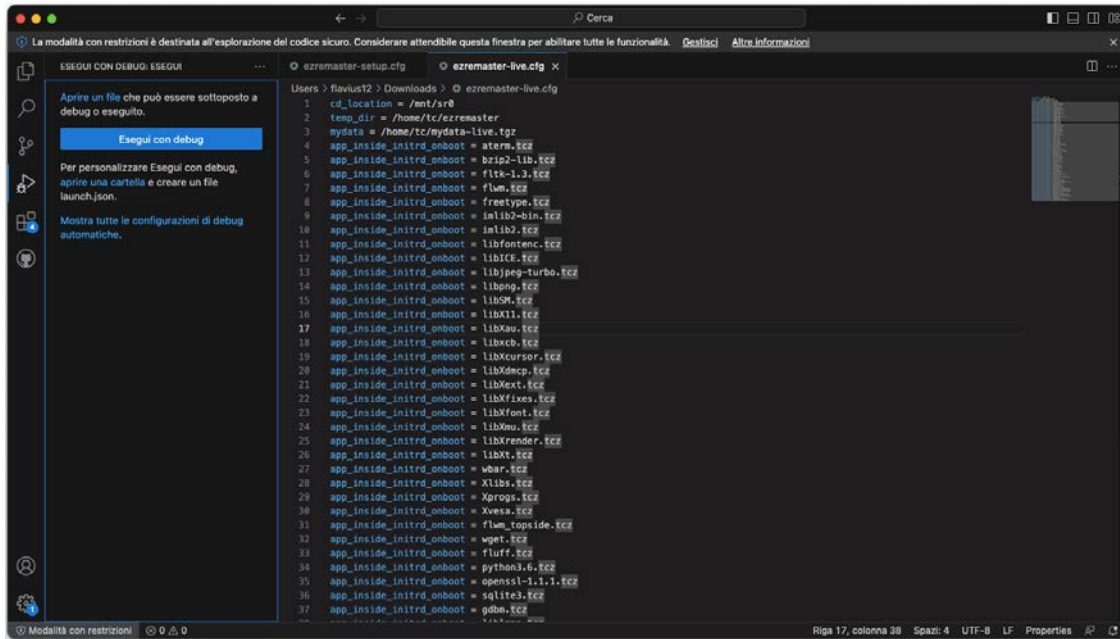
Terminati i passaggi sopra descritti, è possibile avviare la procedura di remastering. Al termine di tale procedura, verrà generato e salvato il file ISO remasterizzato.



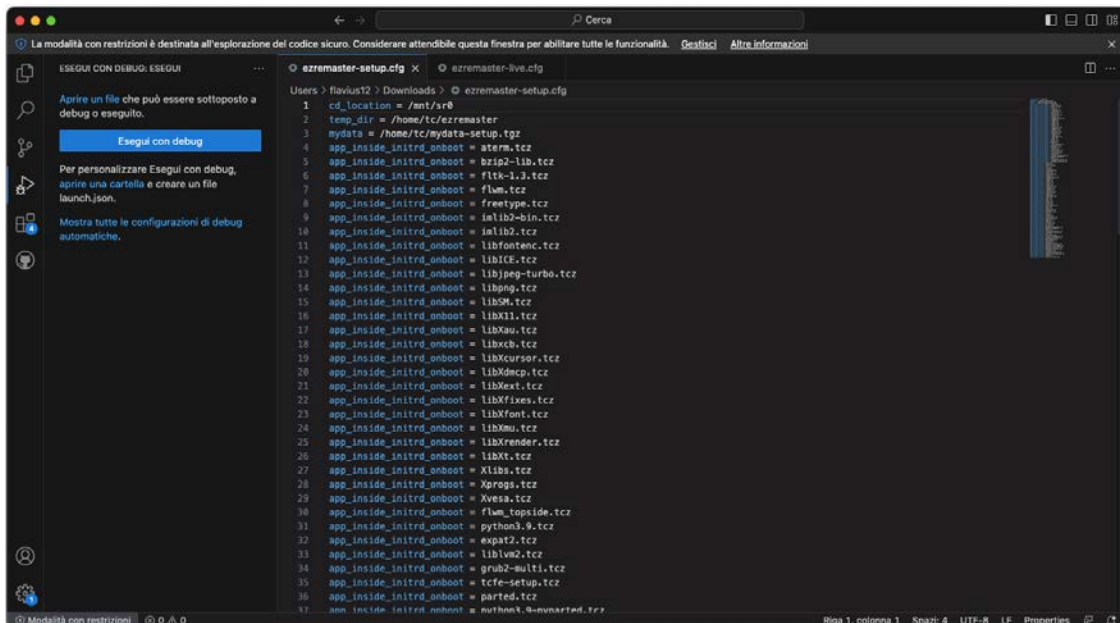
Tuttavia, data l'elevata personalizzazione richiesta da questo lavoro di tesi, il tool EzRemaster verrà usato solo in parte e, principalmente, tramite terminale.

In particolare, sono stati generati due file di configurazione di EzRemaster, `ezremaster-live.cfg` per la modalità Live e `ezremaster-setup.cfg` per la modalità installazione.

Creazione della ISO



```
1 cd_location = /mnt/sr0
2 temp_dir = /home/tc/ezremaster
3 mydata = /home/tc/mydata-live.tgz
4 app_inside_initrd_onboot = aterm.tcz
5 app_inside_initrd_onboot = bzip2-lib.tcz
6 app_inside_initrd_onboot = fltk-1.3.tcz
7 app_inside_initrd_onboot = flwm.tcz
8 app_inside_initrd_onboot = freetype.tcz
9 app_inside_initrd_onboot = intl2-bin.tcz
10 app_inside_initrd_onboot = intl2.tcz
11 app_inside_initrd_onboot = libfontenc.tcz
12 app_inside_initrd_onboot = libICE.tcz
13 app_inside_initrd_onboot = libjpeg-turbo.tcz
14 app_inside_initrd_onboot = libpng.tcz
15 app_inside_initrd_onboot = libSM.tcz
16 app_inside_initrd_onboot = libX11.tcz
17 app_inside_initrd_onboot = libXau.tcz
18 app_inside_initrd_onboot = libxcb.tcz
19 app_inside_initrd_onboot = libXcursor.tcz
20 app_inside_initrd_onboot = libXdmcp.tcz
21 app_inside_initrd_onboot = libXext.tcz
22 app_inside_initrd_onboot = libXfixes.tcz
23 app_inside_initrd_onboot = libXfont.tcz
24 app_inside_initrd_onboot = libXmu.tcz
25 app_inside_initrd_onboot = libXrender.tcz
26 app_inside_initrd_onboot = libXt.tcz
27 app_inside_initrd_onboot = libXi.tcz
28 app_inside_initrd_onboot = libXtst.tcz
29 app_inside_initrd_onboot = libXtst.tcz
30 app_inside_initrd_onboot = libXtst.tcz
31 app_inside_initrd_onboot = libXtst.tcz
32 app_inside_initrd_onboot = libXtst.tcz
33 app_inside_initrd_onboot = libXtst.tcz
34 app_inside_initrd_onboot = libXtst.tcz
35 app_inside_initrd_onboot = libXtst.tcz
36 app_inside_initrd_onboot = libXtst.tcz
37 app_inside_initrd_onboot = libXtst.tcz
```



```
1 cd_location = /mnt/sr0
2 temp_dir = /home/tc/ezremaster
3 mydata = /home/tc/mydata-setup.tgz
4 app_inside_initrd_onboot = aterm.tcz
5 app_inside_initrd_onboot = bzip2-lib.tcz
6 app_inside_initrd_onboot = fltk-1.3.tcz
7 app_inside_initrd_onboot = flwm.tcz
8 app_inside_initrd_onboot = freetype.tcz
9 app_inside_initrd_onboot = intl2-bin.tcz
10 app_inside_initrd_onboot = intl2.tcz
11 app_inside_initrd_onboot = libfontenc.tcz
12 app_inside_initrd_onboot = libICE.tcz
13 app_inside_initrd_onboot = libjpeg-turbo.tcz
14 app_inside_initrd_onboot = libpng.tcz
15 app_inside_initrd_onboot = libSM.tcz
16 app_inside_initrd_onboot = libX11.tcz
17 app_inside_initrd_onboot = libXau.tcz
18 app_inside_initrd_onboot = libxcb.tcz
19 app_inside_initrd_onboot = libXcursor.tcz
20 app_inside_initrd_onboot = libXdmcp.tcz
21 app_inside_initrd_onboot = libXext.tcz
22 app_inside_initrd_onboot = libXfixes.tcz
23 app_inside_initrd_onboot = libXfont.tcz
24 app_inside_initrd_onboot = libXmu.tcz
25 app_inside_initrd_onboot = libXrender.tcz
26 app_inside_initrd_onboot = libXt.tcz
27 app_inside_initrd_onboot = libXi.tcz
28 app_inside_initrd_onboot = libXtst.tcz
29 app_inside_initrd_onboot = libXtst.tcz
30 app_inside_initrd_onboot = libXtst.tcz
31 app_inside_initrd_onboot = libXtst.tcz
32 app_inside_initrd_onboot = libXtst.tcz
33 app_inside_initrd_onboot = libXtst.tcz
34 app_inside_initrd_onboot = libXtst.tcz
35 app_inside_initrd_onboot = libXtst.tcz
36 app_inside_initrd_onboot = libXtst.tcz
37 app_inside_initrd_onboot = libXtst.tcz
```

Il primo file carica come backup il file `mydata-live.tgz` ed imposta tutte le estensioni per essere copiate e caricate all'avvio del sistema.

Il secondo file carica come backup il file `mydata-setup.tgz` ed imposta solo le estensioni necessarie all'avvio dell'installer per essere caricate: le restanti estensioni verranno solo copiate.

Delle estensioni non necessarie fa parte anche la wbar, in quanto nella modalità installazione deve essere avviato solo l'installer: non è necessario dare la possibilità all'utente di avviare ulteriori applicazioni tramite la wbar.

5.4. Personalizzazione di ISOLINUX

ISOLINUX è un boot loader per Linux che opera su CD-ROM ISO 9660/El Torito in modalità "no emulation".²⁹ Esso è utilizzato come bootloader all'interno della ISO di Tiny Core.

EzRemaster, sebbene fornisca numerose personalizzazioni in fase di remastering della ISO, non permette tuttavia la personalizzazione di ISOLINUX.

Per implementare la modalità Live e la modalità installazione all'interno di un'unica ISO è necessario modificare la configurazione di ISOLINUX, in modo da definire nel menu di avvio le due modalità e consentire l'avvio di ognuna di esse.

Dunque è stato definito il file di configurazione di ISOLINUX, denominato `isolinux.cfg`, dal seguente contenuto:

```
DEFAULT tcfe
UI menu.c32
PROMPT 0
TIMEOUT 600
ONTIMEOUT tcfe
F1 f1
F2 f2
F3 f3
F4 f4

MENU TITLE TinyCore Forensic Edition 3.0
MENU MARGIN 10
MENU VSHIFT 5
MENU ROWS 5
MENU TABMSGROW 14
MENU TABMSG Press ENTER to boot, TAB to edit, or press F1 for
more information.
MENU HELPMSGROW 15
MENU HELPMSGENDROW -3
```

²⁹ <https://wiki.syslinux.org/wiki/index.php?title=ISOLINUX>

```
MENU AUTOBOOT BIOS default device boot in # second{,s}...

LABEL tcfe
MENU LABEL TinyCore Forensic Edition Live
TEXT HELP
Boot TinyCore Forensic Edition Live.
Boot media is removable. Use TAB to edit options for specific
needs.
ENDTEXT
KERNEL /boot/vmlinuz
INITRD /boot/core.gz
append loglevel=3 base norestore

LABEL tcfesetup
MENU LABEL Install TinyCore Forensic Edition
TEXT HELP
Install TinyCore Forensic Edition on hard drive.
Boot media is removable. Use TAB to edit options for specific
needs.
ENDTEXT
KERNEL /boot/vmlinuz
INITRD /boot/coresetup.gz
append loglevel=3 base norestore
```

In particolare vengono definiti due elementi del menu: il primo, identificato come `tcfe`, è denominato “TinyCore Foensic Edition Live” mentre il secondo, identificato come `tcfesetup`, è denominato “Install TinyCore Forensic Edition”. Selezionando il primo si avvierà dunque il sistema in modalità Live, mentre selezionando il secondo si avvierà la modalità installazione.

Il titolo del bootloader è impostato su “TinyCore Forensic Edition 3.0” e il menu viene mostrato per 60 secondi, prima di selezionare automaticamente la modalità live.

Nei parametri forniti al kernel per l’avvio, in entrambe le modalità sono specificati il parametro `base` e il parametro `norestore`: il primo serve a prevenire il caricamento di dati da altre installazioni di Tiny Core presenti su eventuali dischi rigidi (in tal modo viene effettuato un avvio pulito del sistema, caricando esclusivamente ciò che è all’interno del `core.gz`), il

secondo previene il caricamento di backup (file `mydata.tgz`) presenti su eventuali dischi rigidi.

5.5. Script per la creazione della ISO finale (`makeiso.sh`)

Per generare la ISO finale è stato sviluppato un piccolo script, denominato `makeiso.sh`:

```
#bin/sh
cp -f isolinux.cfg image/boot/isolinux/
cd image
chmod 444 boot/isolinux/isolinux.cfg
#rm cde/optional/wbar.*
mkisofs -l -J -R -V TCFE -no-emul-boot -boot-load-size 4 -
boot-info-table -b boot/isolinux/isolinux.bin -c
boot/isolinux/boot.cat -o /home/tc/ezremaster/ezremaster.iso .
```

Questo script si occupa di copiare innanzitutto la configurazione personalizzata di ISOLINUX, cambiando anche i permessi del file di configurazione ed infine genera la ISO con l'utility `mkisofs`.

In particolare, per generare la ISO finale è necessario eseguire, in ordine, le seguenti operazioni:

```
$ remaster.sh /home/tc/ezremaster/ezremaster-setup.cfg rebuild
```

Recarsi nella cartella `/home/tc/ezremaster/image/boot` e rinominare il file `core.gz` in `coresetup.gz`:

```
$ cd /home/tc/ezremaster/image/boot
$ sudo mv core.gz coresetup.gz
```

Effettuare il remaster per la versione live:

```
$ remaster.sh /home/tc/ezremaster/ezremaster-live.cfg rebuild
```

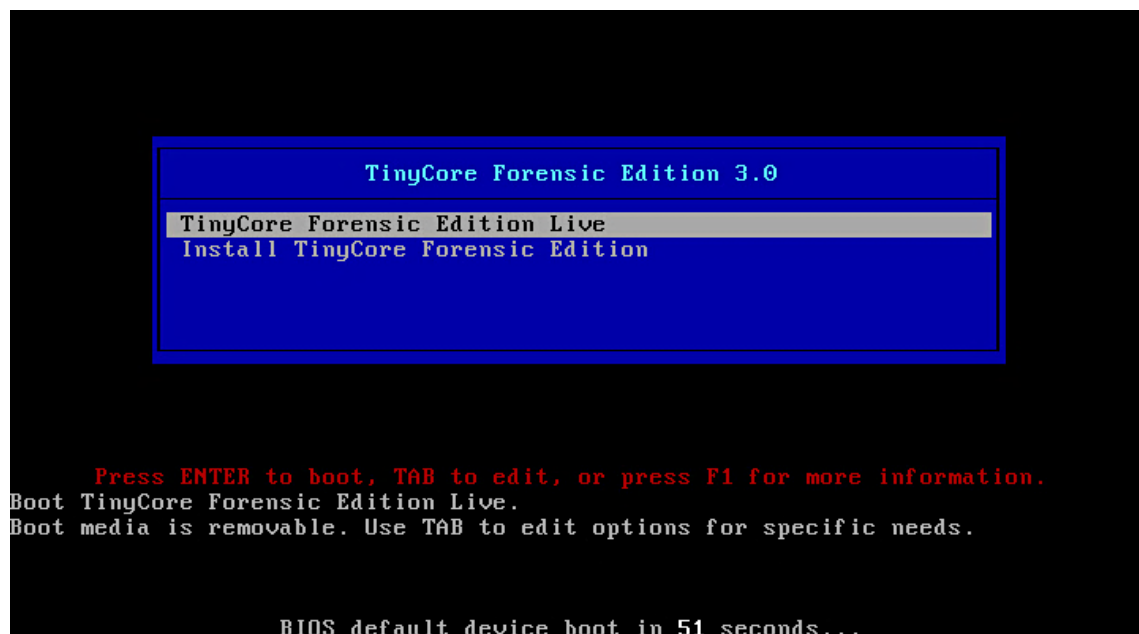
Eseguire makeiso.sh:

```
$ cd /home/tc/ezremaster  
$ sudo makeiso.sh
```

Al termine dell'esecuzione verrà generato il file ISO finale denominato `ezremaster.iso` e posizionato in `/home/tc/ezremaster`. Tale file viene infine rinominato in `Tiny.Core.Forensic.Edition-v3.0.iso`.

6. Testing

Una volta generato il prodotto finale, si procede alla fase di testing. In particolare si testa sia la modalità Live, verificando la conformità con le funzionalità presenti nelle versioni precedenti, sia la modalità installazione, verificando che al termine dell'installazione del sistema, esso sia avviabile correttamente da disco rigido. All'avvio della ISO viene mostrato il seguente menu di boot:

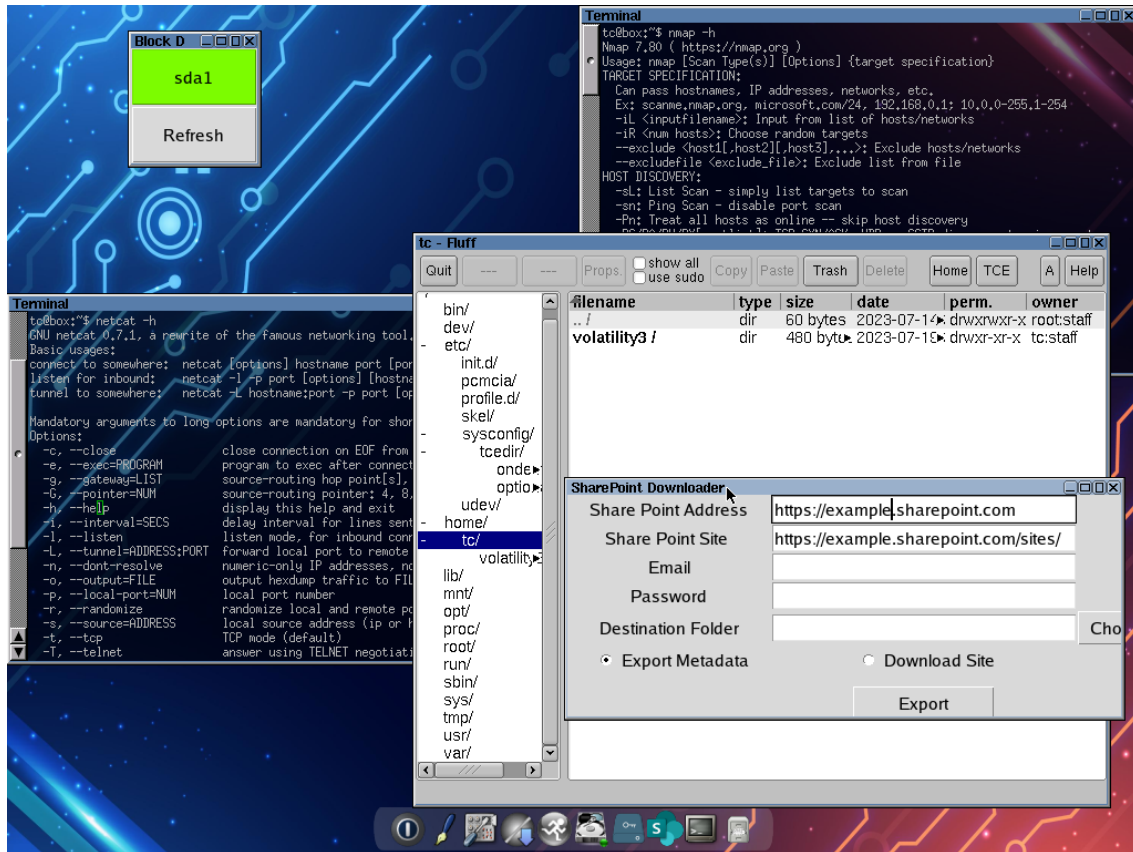


6.1. Esecuzione della modalità Live

All'avvio della modalità Live, viene correttamente caricato il Write Blocker e successivamente il desktop:



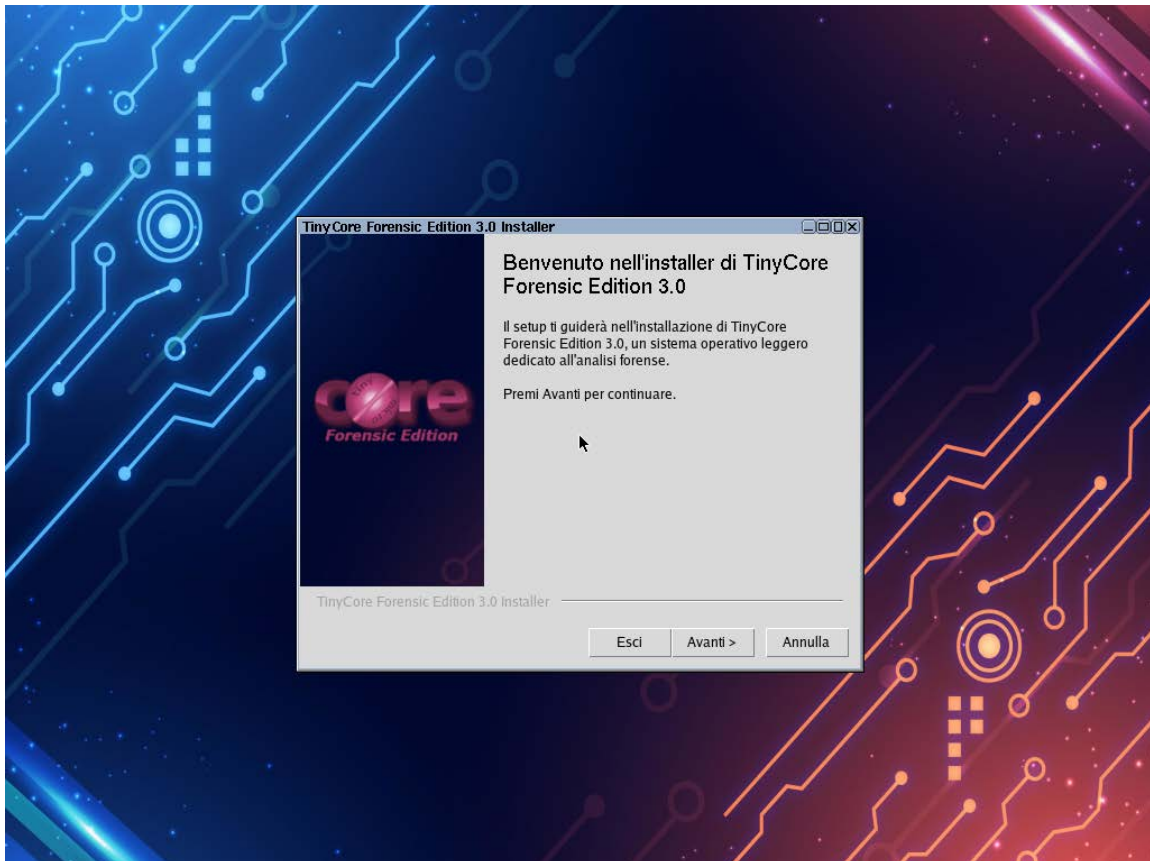
Ora vengono avviati i vari tool forensi installati, in particolare: BlockDevGui, SharePoint Downloader, nmap e netcat, insieme al file manager fluff:



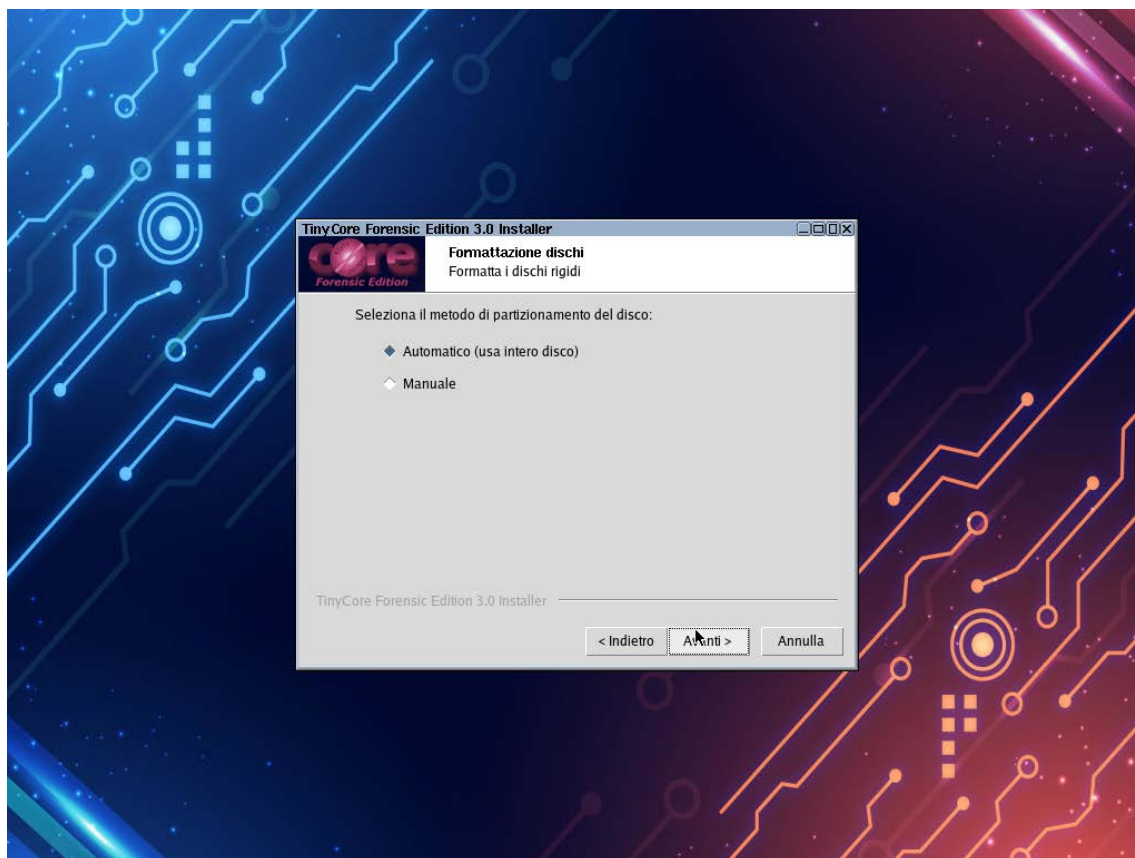
6.2. Esecuzione dell'installer

6.2.1. Test 1: Partizionamento automatico

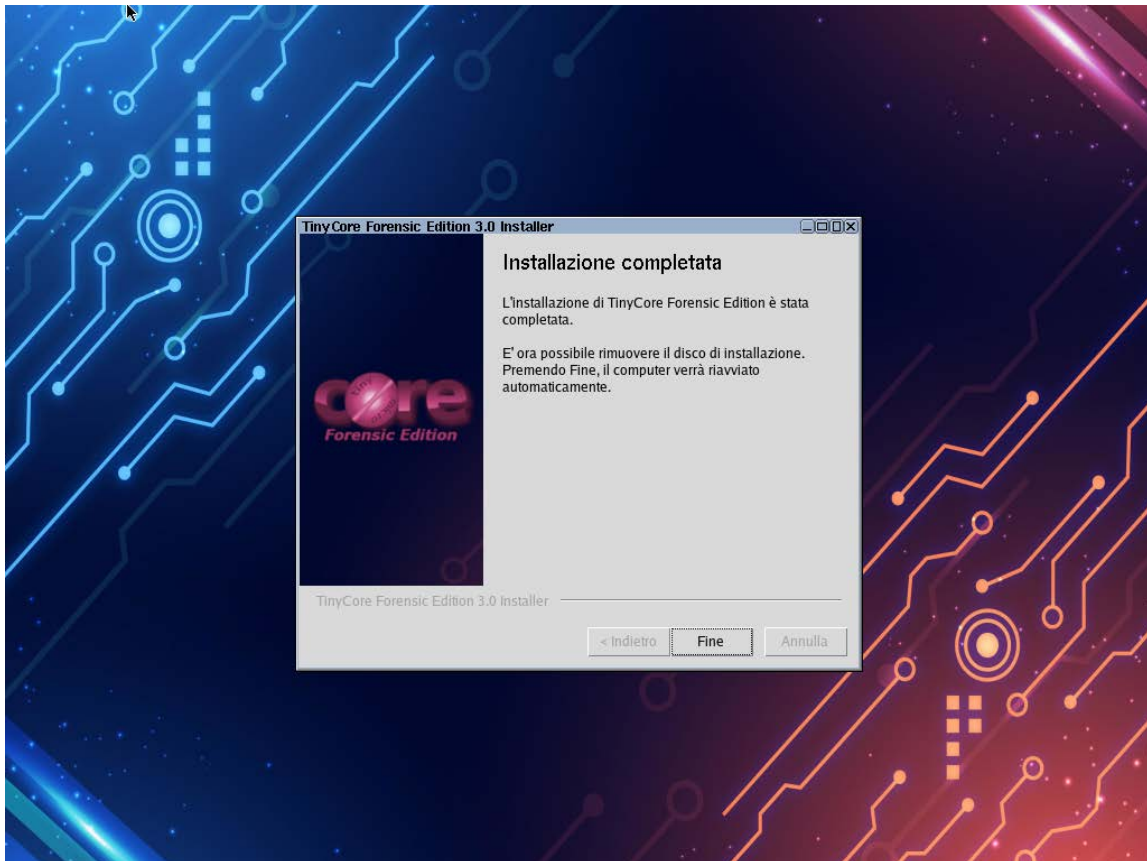
Per la modalità installazione vengono testati sia il partizionamento automatico che manuale. Dapprima si avvia la modalità installazione dalla ISO, verificando che l'installer si avvii automaticamente:



Premendo “Avanti >” viene mostrato il frame in cui è possibile selezionare il metodo di partizionamento. Come anticipato, nel primo test, è stato selezionato il partizionamento automatico:



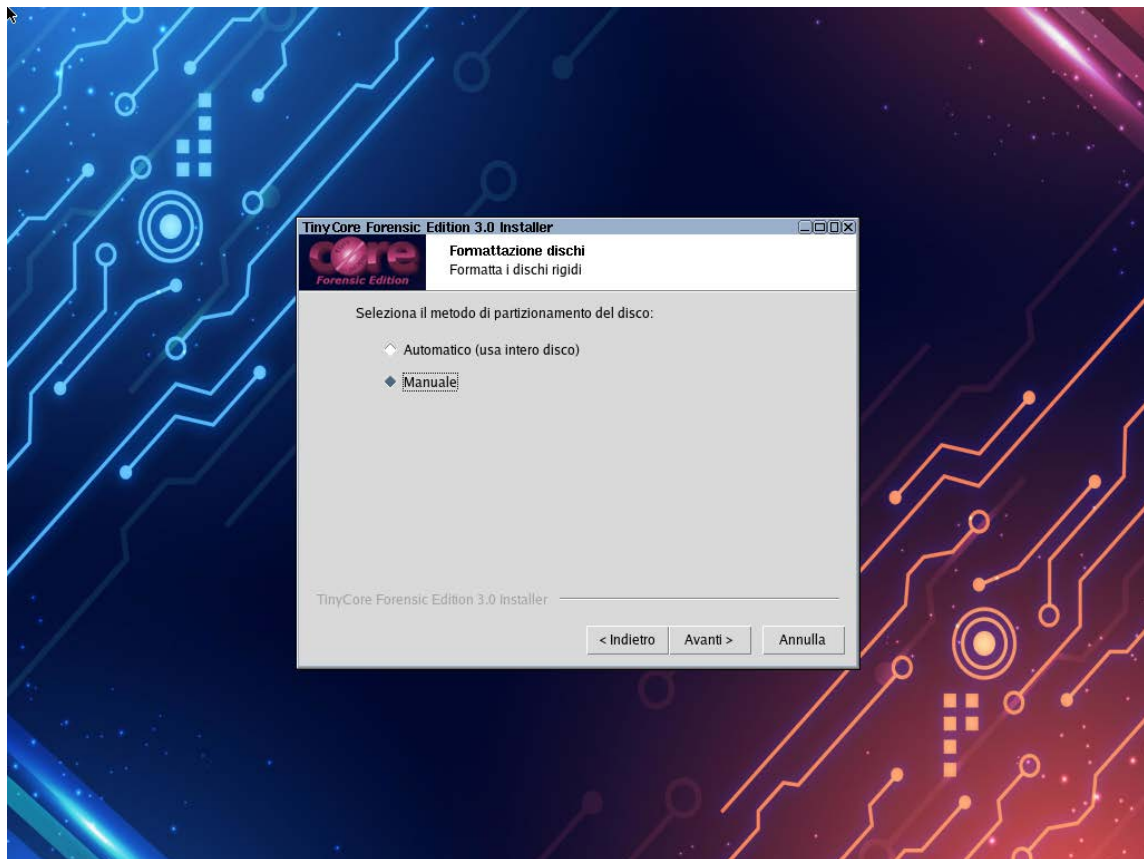
I file vengono copiati correttamente e viene mostrata la schermata di completamento dell'installazione:



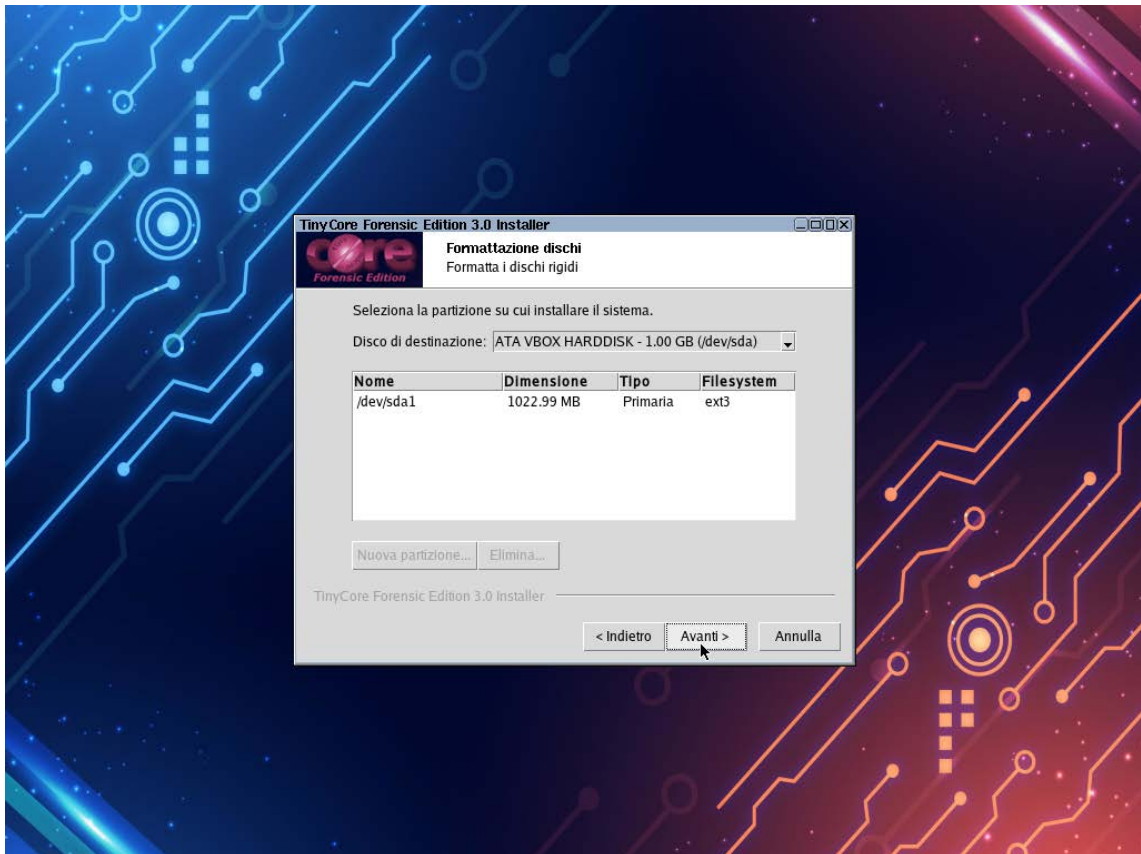
Rimuovendo la ISO e riavviando il sistema, esso si avvia correttamente.

6.2.2. Test 2: Partizionamento manuale

Si ripete lo stesso test precedente, ora selezionando il partizionamento manuale:

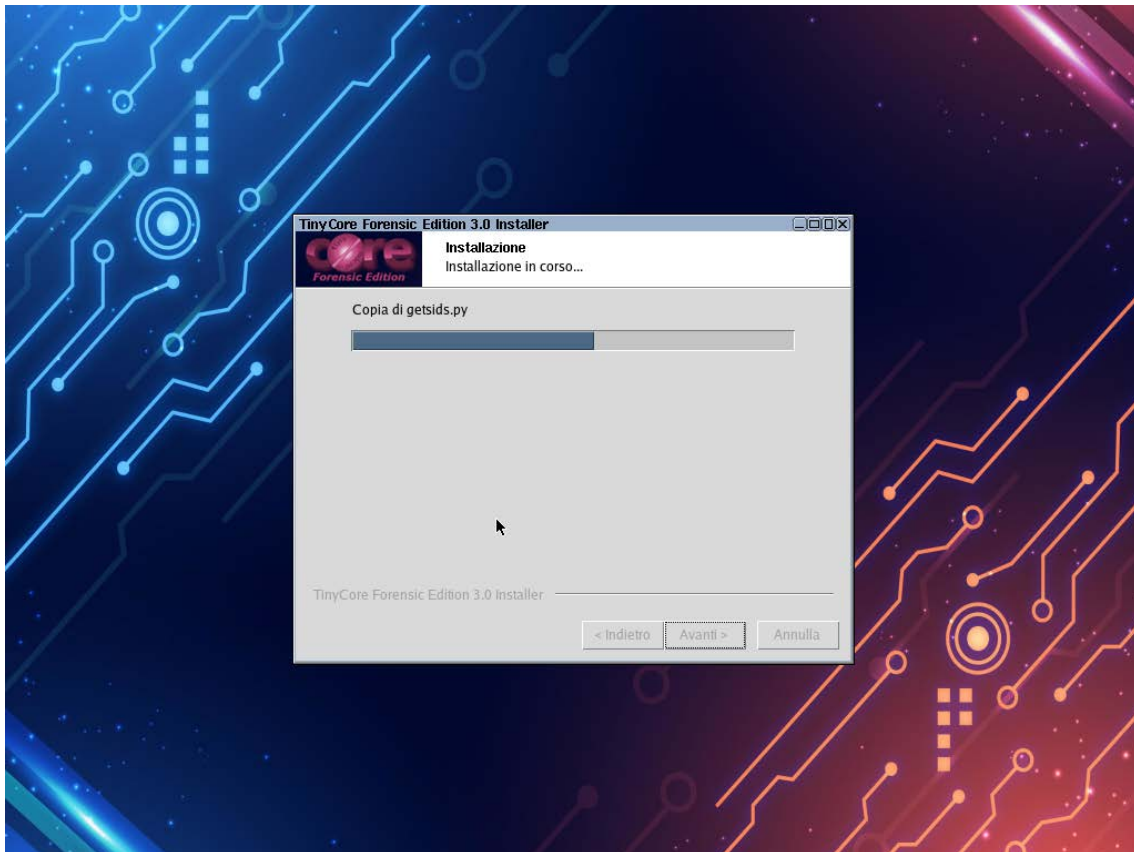


Premendo “Avanti >”, viene mostrata la schermata di partizionamento:



Si nota la presenza della partizione creata in precedenza, nella quale risiede il sistema operativo. Ora si procede con l'eliminazione della vecchia partizione e se ne crea una nuova su cui verrà installato il sistema.

Completata l'operazione, si procede con l'installazione:

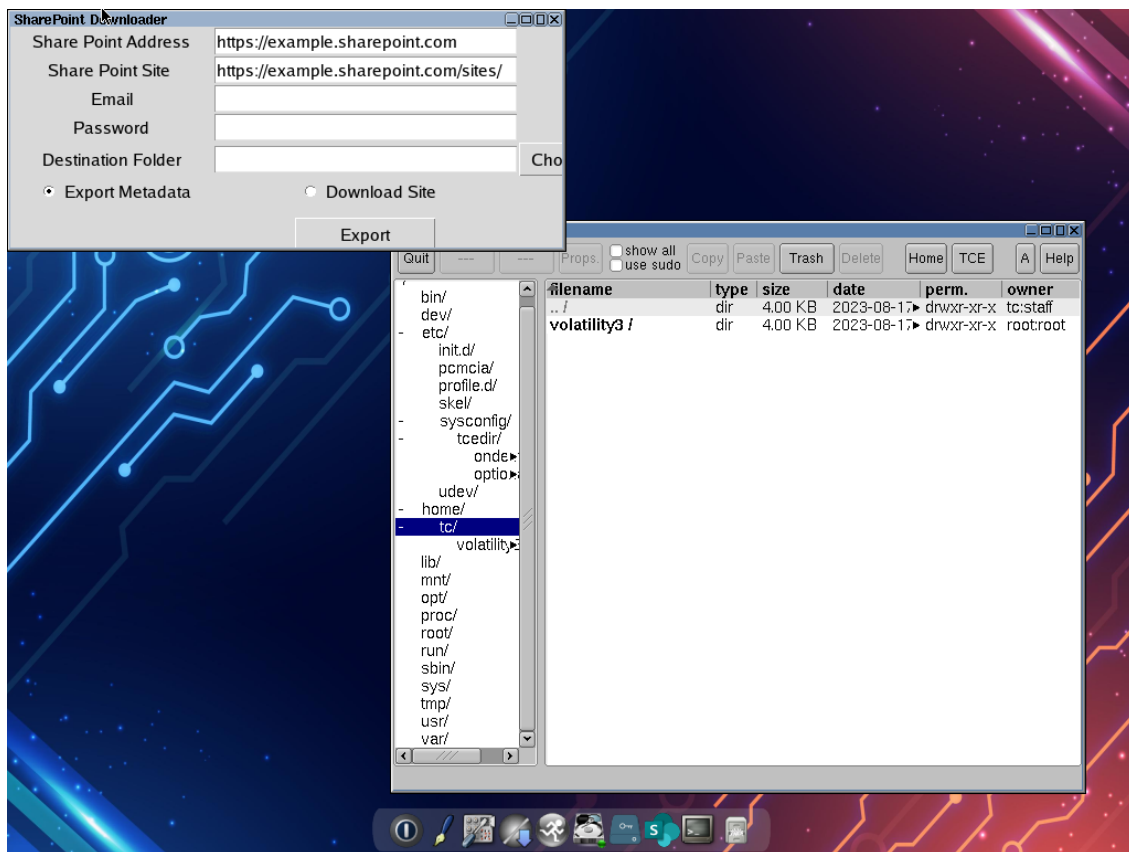


Anche in questo caso l'installazione viene completata correttamente. Ora si procede alla rimozione della ISO e al riavvio dell'host.

Al riavvio il sistema viene caricato correttamente: viene mostrata la schermata del bootloader GRUB:



Premendo Invio, viene correttamente caricato il desktop:



7. Risultati

Il prodotto finale di questo lavoro di tesi è la generazione del file ISO `Tiny.Core.Forensic.Edition-v3.0.iso`. Tale file conterrà sia la versione Live di Tiny Core Forensic Edition, sia l'installer che permette di installare Tiny Core Forensic Edition su disco rigido. La dimensione finale della ISO è di 367 MB.

Il file ISO finale e le ulteriori risorse correlate sono scaricabili al seguente URL: <https://github.com/Flavius12/TinyCore-FE-Installer/releases>

8. Sviluppi futuri

L'obiettivo alla base di questo lavoro di tesi è stato quello di ampliare il progetto di Tiny Core Forensic Edition sviluppato dagli studenti Ruffo S. e De Benedittis F., aggiungendo un installer che permettesse l'installazione del sistema su disco rigido. Ulteriore obiettivo è stato quello di aggiornare il Core alla base della distribuzione, in modo da includere nuove feature, bug fix e patch di sicurezza sia a livello di kernel Linux, sia a livello di Tiny Core.

Tuttavia, possibili sviluppi futuri potrebbero consistere nell'aggiungere ulteriori tool forensi, integrando ad esempio un tool di acquisizione forense grafico e sviluppare anche un Desktop Manager più completo e User Friendly, dotandolo di un'interfaccia grafica più completa, stilizzata e funzionale.

9. Bibliografia

- [1] *Linux man page, «ewfacquire(1): acquires data in EWF format,»*
[Online]. Available: <https://linux.die.net/man/1/ewfacquire>.
- [2] *Wikipedia, «CAINE Linux,»* [Online]. Available:
https://en.wikipedia.org/wiki/CAINE_Linux. [Consultato il giorno 20
Agosto 2023].
- [3] *R. Sara, Analisi, Progettazione e Realizzazione di Tiny Core Forensic
Edition.*
- [4] *F. De Benedittis, Realizzazione di funzionalità avanzate in Tiny Core
Forensic Edition.*
- [5] *Wikipedia, «Foremost (software),»* [Online]. Available:
[https://en.wikipedia.org/wiki/Foremost_\(software\)](https://en.wikipedia.org/wiki/Foremost_(software)).
- [6] *Wikipedia, «Nmap,»* [Online]. Available:
<https://it.wikipedia.org/wiki/Nmap>.
- [7] *Wikipedia, «Netcat,»* [Online]. Available:
<https://it.wikipedia.org/wiki/Netcat>.
- [8] *rhash, «rhash/RHash: Great utility for computing hash sums,»*
[Online]. Available: <https://github.com/rhash/RHash>.
- [9] *Wikipedia, «The Sleuth Kit,»* [Online]. Available:
https://en.wikipedia.org/wiki/The_Sleuth_Kit.
- [10] *volatilityfoundation, «volatilityfoundation/volatility3: Volatility 3.0
development,»* [Online]. Available:
<https://github.com/volatilityfoundation/volatility3>.

- [11] Python Software Foundation, «tkinter — Python interface to Tcl/Tk,» [Online]. Available: <https://docs.python.org/3/library/tkinter.html>.
- [12] Wikipedia, «Python Imaging Library,» [Online]. Available: https://en.wikipedia.org/wiki/Python_Imaging_Library.
- [13] dcantrell, «dcantrell/pyparted: Python bindings for GNU parted (libparted),» [Online]. Available: <https://github.com/dcantrell/pyparted>.
- [14] Wikipedia, «Termcap,» [Online]. Available: <https://en.wikipedia.org/wiki/Termcap>.
- [15] alejandroautalan, «alejandroautalan/pygubu-designer: A simple GUI designer for the python tkinter module,» [Online]. Available: <https://github.com/alejandroautalan/pygubu-designer>.
- [16] Syslinux, «ISOLINUX - Syslinux Wiki,» [Online]. Available: <https://wiki.syslinux.org/wiki/index.php?title=ISOLINUX>.
- [17] FBI, «Piecing Together Digital Evidence,» 8 Gennaio 2013. [Online]. Available: <https://www.fbi.gov/news/stories/piecing-together-digital-evidence>.
- [18] Wikipedia, «Scienza forense,» [Online]. Available: https://it.wikipedia.org/wiki/Scienza_forense.
- [19] Wikipedia, «Digital forensics,» [Online]. Available: https://it.wikipedia.org/wiki/Digital_forensics.
- [20] Wikipedia, «Informatica forense,» [Online]. Available: https://it.wikipedia.org/wiki/Informatica_forense.
- [21] CAINE Linux, «CAINE Linux,» [Online]. Available: <https://www.caine-live.net>.

- [22] R. Berti, P. Dal Checco e F. Zumerle, «Distribuzioni forensi per Linux: a cosa servono e le migliori per le analisi forensi,» [Online]. Available: <https://www.cybersecurity360.it/soluzioni-aziendali/distribuzioni-forensi-per-linux-a-cosa-servono-e-le-migliori-per-le-analisi-forensi/>.
- [23] Tsurugi Linux, «Tsurugi Linux | Digital Forensics, Osint and malware analysis Linux Distribution,» [Online]. Available: <https://tsurugi-linux.org>.
- [24] GeeksforGeeks, «'dd' command in Linux,» [Online]. Available: <https://www.geeksforgeeks.org/dd-command-linux/>.
- [25] GNU - Free Software Foundation, «Ddrescue - Strumento di recupero dati,» [Online]. Available: https://www.gnu.org/software/ddrescue/ddrescue_it.html);
- [26] GNU - Free Software Foundation, «Parted - GNU Project - Free Software Foundation,» [Online]. Available: <https://www.gnu.org/software/parted/>.