UNIVERZA V LJUBLJANI
FRI
Fakulteta za računalništvo in informatiko

Špela Koračin in Svit Jesenk

Varnostna kopija datotek in diska

Seminar pri predmetu Sistemska programska oprema

MENTOR: doc. dr. Tomaž Dobravec

Ljubljana, januar 2025

Povzetek

Seminar preučuje ključne vidike varnostnega kopiranja podatkov, vključno s tehnologijami, kot so fizične kopije, oblačne storitve, šifriranje in deduplikacija, ter različne metode, kot so popolno, diferencialno in inkrementalno varnostno kopiranje. Poudarja tudi specifike posameznih operacijskih sistemov, vključno z Windows, Linux, macOS, Android in iOS, ter načine optimizacije prostora in izboljšanja varnosti podatkov. Predstavljen je primer implementacije na macOS z uporabo Python skripte, ki vključuje funkcionalnosti za ustvarjanje snapshot-ov, stisnjeno varnostno kopiranje in preverjanje integritete podatkov. Poglobljen pregled tehnik in orodij omogoča učinkovito načrtovanje varnostnega kopiranja za različne potrebe.

Kazalo vsebine

Uvod

Varnostno kopiranje podatkov je eden najpomembnejših procesov za zagotavljanje integritete in dostopnosti podatkov v sodobnem digitalnem svetu. Izguba podatkov zaradi tehničnih okvar, kibernetskih napadov ali drugih nepredvidenih dogodkov lahko prinese nepopravljivo škodo. V seminarju sva raziskala različne pristope, tehnologije in algoritme za varnostno kopiranje ter njihove specifike in uporabo na različnih operacijskih sistemih. Namen je poglobiti razumevanje delovanja teh procesov in ponuditi smernice za implementacijo učinkovitih rešitev za zaščito podatkov.

Sistemsko neodvisna logika

Varnostno kopiranje podatkov

Varnostno kopiranje podatkov je postopek kopiranja podatkov v IT sistemu na drugo lokacijo, tako da jih je mogoče obnoviti, če se izvirni podatki izgubijo. Namen postopka varnostnega kopiranja je ohraniti podatke v primeru okvare opreme, kibernetskega napada, naravne katastrofe ali drugih dogodkov, ki povzročijo izgube podatkov.

Katere podatke in kako pogosto

Postopek varnostnega kopiranja je mogoče uporabiti za podatke, shranjene v različnih okoljih. Ti vključujejo fizične strežnike, baze podatkov, omrežno shrambo, virtualne stroje, javno shrambo v oblaku, aplikacije v kontejnerjih, aplikacije SaaS in končne točke, kot so prenosniki in mobilne naprave.
Pogostost popolnega varnostnega kopiranja podatkov, ki zajema vse datoteke, ki jih želi podjetje ali fizična oseba zaščititi, je odvisna od faktorjev, kot sta kritičnost podatkov in kako pogosto se spreminjajo. Torej je popolna varnostna kopija lahko načrtovana dnevno, tedensko ali v kakšnem drugem intervalu.

Tehnologije

Fizične kopije

Diski (HDD ali SSD)
Večina današnjih podatkov je varnostno kopiranih na trdi disk (HDD) ne glede na to, ali je ta pogon samostojen zunanji pogon ali del rezervnega strežnika.

SDD-ji ponujajo hitrejše čase branja/pisanja kot trdi diski, zahtevajo manj fizičnega prostora za shranjevanje enake količine podatkov in porabijo manj energije, čeprav na gigabajt prostora le-ti dražji. Če imajo trdi diski in diski SDD pomanjkljivost, je to, da ne ponujajo razširljivosti – če potrebujete več zmogljivosti za varnostno kopiranje, morate kupiti in namestiti nov fizični disk.

Strežniki
Strežnik za varnostno kopiranje je namenski strežnik, zgrajen posebej za varnostno kopiranje datotek, shranjenih v več odjemalskih računalnikih v istem omrežju. Strežnik je opremljen s pomembnim diskovnim pomnilnikom in specializirano programsko opremo za načrtovanje in upravljanje varnostnih kopij.

Diski strežnika za varnostno kopiranje so pogosto konfigurirani za redundanco, da zaščitijo varnostne kopije podatkov in zagotovijo, da se varnostne kopije nadaljujejo v primeru okvare diska.Varnostnih kopij podatkov ne ščiti pred lokalnimi izpadi ali fizičnimi nesrečami.

Oblačne kopije

Storitve varnostnega kopiranja v oblaku pogosto varnostno kopirajo vaše podatke na oddaljene strežnike. Ta postopek vključuje datoteke, dokumente, slike, filme, zbirke podatkov in še več.
Ponudnik storitev v oblaku shranjuje podatke v varnih podatkovnih centrih. Ti podatkovni centri zagotavljajo močno varnost in redundanco za celovitost in razpoložljivost podatkov.

Načini varnostnega kopiranja

Popolno varnostno kopiranje

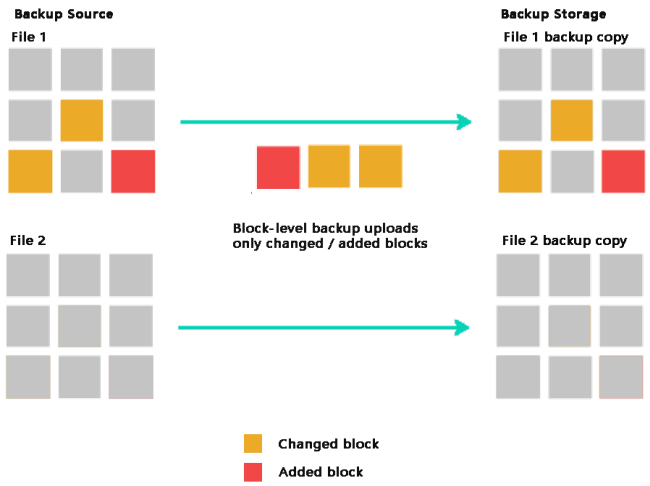
Popolna varnostna kopija je najbolj popolna vrsta varnostne kopije, kjer se klonirajo vsi izbrani podatki. To vključuje datoteke, mape, aplikacije SaaS, trde diske in še več. Ker je vse varnostno kopirano naenkrat, varnostno kopiranje traja dlje v primerjavi z drugimi vrstami varnostnih kopij.
Druga pogosta težava pri izvajanju popolnih varnostnih kopij je, da preobremenjuje prostor za shranjevanje. Zato večina podjetij ponavadi izvaja popolno varnostno kopiranje in občasno sledi diferencialnemu ali postopnemu varnostnemu kopiranju. To zmanjšuje obremenitev prostora za shranjevanje in povečuje hitrost varnostnega kopiranja.

Inkrementalno

Prva varnostna kopija v inkrementalni varnostni kopiji je popolna varnostna kopija. Vsaka naslednja varnostna kopija pa shrani le spremembe, ki so bile narejene v času od prejšnje varnostne kopije. Shranjene so le najnovejše spremembe.
Inkrementalno varnostno kopiranje zahteva prostor le za shranjevanje sprememb, kar omogoča hitro varnostno kopiranje.

- BLOČNO

Varnostno kopiranje na ravni blokov je funkcija inkrementalnega varnostnega kopiranja, ki omogoča nalaganje samo spremenjenih in dodanih blokov namesto celotnih datotek. Za se to uporablja tehnologija posnetkov (snapshot), pri čemer gre za trenutno sliko sistema, ki predstavlja točno tako stanje, kot je bilo v trenutku posnetka. Programska oprema prebere podatke v blokih enake velikosti, izračuna zgoščene vrednosti ločeno za vsak blok in jih nato primerja s prejšnjo vrednostjo. Če pride do neusklajenosti, bo blok naložen v shrambo za varnostne kopije.



bločno kopiranje

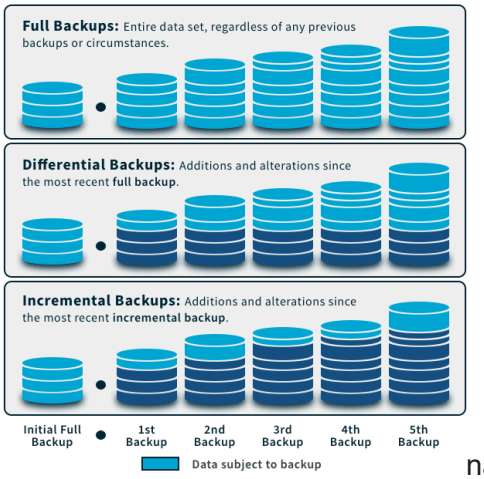
- DATOTEČNO

Ko se izvede inkrementalno varnostno kopiranje na ravni datoteke in je datoteka spremenjena, se celotna datoteka kopira v skladišče varnostnih kopij.

Diferencialno

Ta vrsta varnostnega kopiranja vključuje varnostno kopiranje podatkov, ki so bili ustvarjeni ali spremenjeni od zadnje popolne varnostne kopije. Na začetku se izvede popolna varnostna kopija, nato pa se zaženejo nadaljnje varnostne kopije, ki vključujejo vse spremembe datotek in map.

Omogoča hitrejše obnavljanje podatkov kot popolno varnostno kopiranje, saj zahteva le dve komponenti varnostne kopije: začetno popolno varnostno kopiranje in najnovejšo diferencialno varnostno kopijo.



načini varnostnega kopiranja

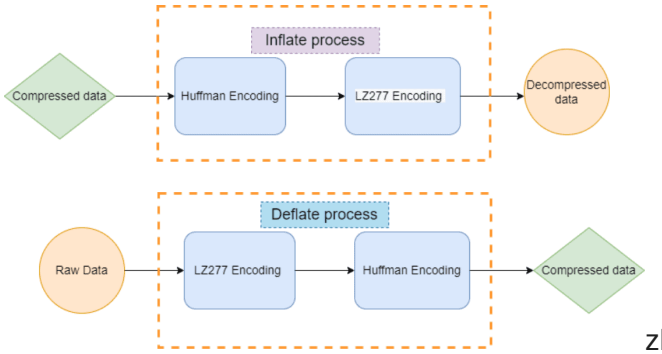
Zmanjšanje obsega podatkov

Tehnike zmanjševanja podatkov predstavljajo rešitev za težave s shranjevanjem tako, da zagotavljajo način za zmanjšanje podatkov brez izgube ključnih informacij ali celovitosti. S temi tehnikami lahko ljudje, podjetja in organizacije varnostno kopirajo podatke v kompaktni

obliki in zmanjšajo porabo pasovne širine, hkrati pa zagotovijo, da lahko shranjene informacije pridobijo brez izgube.

Stiskanje podatkov (ZIP, GZIP)

Stiskanje zmanjša velikost podatkovnih datotek tako, da jih kodira ali spremeni, da zmanjša velikost izvirne datoteke, zaradi česar so manjše in bolj kompaktne. Za razliko od deduplikacije, ki deluje na ravni bloka, stiskanje deluje na ravni datoteke. Za stiskanje datoteke algoritem identificira podvojene ali nepotrebne informacije, tako da ugotovi dele, ki jih lahko odpravi, ne da bi pri tem ogrozil kakovost izvirnih informacij. Odvečni podatki bodo nato odstranjeni, ostali pa preurejeni.



zlib algoritem

Metode stiskanja

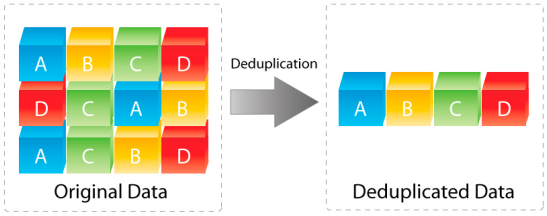
Datoteke lahko stisnete na dva načina:

Stiskanje z izgubo
Stiskanje z izgubo zmanjša velikost datotek z odstranitvijo nepomembnih delov večpredstavnostnih datotek. Zvočni posnetek se lahko na primer skrči v format MP3, ki shranjuje velike zvočne datoteke v nekaj MB. Neslišne zvočne frekvence in drugi neključni elementi bodo med pretvorbo odstranjeni. Torej, čeprav pride do izgube zvestobe zvoka, še vedno ponuja sprejemljivo kakovost zvoka.

Stiskanje brez izgub
Metoda stiskanja brez izgub prav tako odstrani informacije iz datoteke. Stisne datoteke tako, da poišče odvečnosti in jih nato shrani za referenco. Na ta način lahko še vedno rekonstruirate stisnjeno datoteko, da dobite izvirno obliko zapisa. Deluje tako, da preveri datoteko in predstavi ponavljajoče se podatke z označbo mesta za prepoznavanje. Vse odvečne datoteke spadajo pod en identifikator, kar zmanjšuje velikost datoteke pri varnostnem kopiranju. Postopek stiskanja brez izgub se uporablja predvsem za varnostno kopiranje podatkov, ker lahko izguba malo podatkov vpliva na njegovo celovitost. Uporablja se tudi za ustvarjanje datotek zip, ki jih uporabnik lahko pridobi.

Deduplikacija

Deduplikacija podatkov vključuje odstranitev enakih podatkov, da se zmanjša pomnilnik, potreben za shranjevanje velikega kosa podatkov. Na splošno tehnika deduplikacije podatkov identificira in odpravi podvojene podatkovne bloke s kriptografsko zgoščevalno funkcijo. Metode deduplikacije podatkov, ki temeljijo na zgoščevanju, uporabljajo zgoščevalni algoritem za razlikovanje »kosov« podatkov. Pogosto uporabljeni algoritmi so SHA-1 in MD5. Ko zgoščevalni algoritem obdeluje podatke, se ustvari zgoščena (hash) vrednost, ki predstavlja podatke in zazna podvojene z določenimi oblikami primerjalnega postopka. Če isti podatki večkrat preidejo skozi algoritem zgoščevanja, se vsakič ustvari ista vrednost.
Vsak kos je nato indeksiran z identifikacijo, tako da bodo uporabniki lahko po potrebi rekonstruirali podatke.



deduplikacija

Varnost podatkov - enkripcija (AES)

Šifrirana varnostna kopija je zaščitena s šifrirnimi algoritmi, da se ohrani pristnost, zaupnost in celovitost informacij ter prepreči nepooblaščen dostop. Nešifrirana varnostna kopija preprosto pomeni, da shranjeni podatki ali informacije niso kodirane z nobenim algoritmom. Šifrirane varnostne kopije so zavarovane s kompleksnimi algoritmi in so berljive samo tistim uporabnikom s ključem. Nešifrirana varnostna kopija je ranljiva za spletne kršitve in kibernetske napade, in ker je v nezavarovani obliki ali odprtem besedilu, si lahko informacije enostavno ogledate ali dostopate.

Šifriranje med prenosom

To pomeni šifriranje podatkov, ko so v gibanju med napravami in omrežji ali se prenašajo v oblak. Šifriranje med prenosom poteka med virom varnostne kopije (računalnik, strežnik, Salesforce, Microsoft 365, Google Workspace itd.) in ciljem varnostne kopije (Unitrends Cloud, Spanning-managed storage v S3, shramba, ki jo upravlja stranka, če naštejemo le nekatere).

Šifriranje v mirovanju

To pomeni šifriranje podatkov, ko se nahajajo v pomnilniku »v mirovanju« ali na cilju varnostne kopije.

Najpogostejši uporabljen algoritem za šifriranje podatkov je AES (Advanced Encryption Standard). AES je simetrična metoda šifriranja, kar pomeni, da se isti ključ uporablja za

šifriranje in dešifriranje podatkov. Podatke razdeli v bloke po 128 bitov. Bloki gredo skozi več krogov različnih vrst transformacij, kot so zamenjava, premikanje in transpozicija.
AES uporablja dolžine ključev 128, 192 in 256 bitov, pri čemer je 256 najmočnejši.

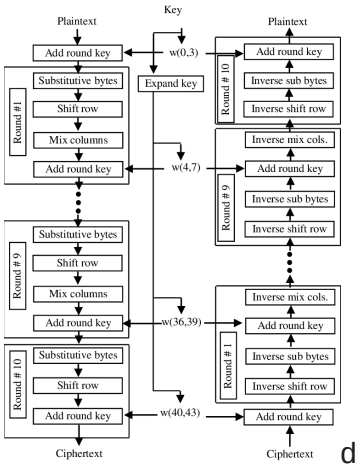
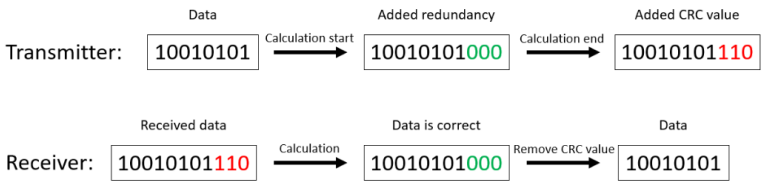


diagram delovanja algoritma AES

Bring Your Own Key (BYOK) je model šifriranja, ki strankam omogoča uporabo lastne programske opreme za šifriranje in ključev za šifriranje in dešifriranje podatkov, shranjenih v oblaku. To vam omogoča večji nadzor nad podatki in upravljanjem ključev. BYOK vašim zaupnim podatkom doda dodatno raven varnosti.

Integriteta podatkov (CRC)

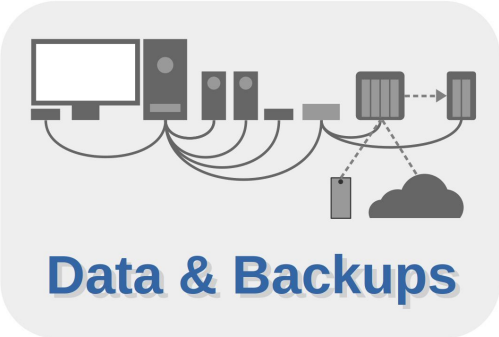
CRC (Cyclic redundancy check) preverja celovitost podatkov, ki so zavarovani z varnostnimi kopijami in preneseni prek omrežja, ter podatkov, shranjenih na mediju.
CRC deluje tako, da podatke obravnava kot binarni polinom. Podatki so deljeni z vnaprej določenim deliteljem, znanim kot generatorski polinom. Preostali del te delitve je kontrolna vsota. Za preverjanje celovitosti podatkov sprejemnik izvede enako operacijo delitve. Če je preostanek enak nič, se podatki štejejo za brez napak. Če je preostanek drugačen od nič, to pomeni, da so bile v podatkih odkrite napake.
CRC ne more popraviti napak. Njegov glavni namen je odkrivanje napak pri prenosu podatkov, ne pa njihovo odpravljanje. Ko se odkrijejo napake, lahko prejemnik od pošiljatelja zahteva, da ponovno prenese podatke, da zagotovi komunikacijo brez napak.
CRC se pri varnostnem kopiranju uporablja v različnih fazah: med prenosom podatkov iz izvornega sistema v ciljni sistem, pri zapisovanju podatkov na fizične medije, kot so diski ali optični mediji ter pri preverjanju integritete arhiviranih podatkov pred obnovitvijo.



CRC algoritem

Specifika za operacijske sisteme

Načini in tehnologije varnostnega kopiranja se precej razlikujejo od enega do drugega operacijskega sistema. Vključen je širok spekter orodij in pristopov, ki so prilagojeni specifikam posameznih platform. Predstavil bom tehnologije za varnostno kopiranje na Windows, Linux, MacOS, in mobilnih sistemih (Android, iOS).



Specifika za Windows

Ko pride do operacijskega sistema Windows moremo prvo govoriti o:

- **Windows Backup & Restore**, ki je privzeto orodje operacijskega sistema Windows za ustvarjanje in upravljanje varnostnih kopij in obnovo podatkov. Prvič je bil predstavljen v Windows Vista, ostal pa je pomemben del Windows okolja vse do sedaj na Windows 11. Ogrodje je vgrajeno v OS Windows in zato ne zahteva dodatne programske opreme. Za preverjanje, če je prišlo do spremembe datoteke, Windows uporablja hash funkcije, kot je SHA-1, ki generira hash vrednost na osnovi njene vsebine. Če se spremeni hash vrednost, se datoteko označi kot spremenjeno.
  Temelji na inkrementalnih in diferencialnih varnostnih kopijah. Ob vsaki spremembi datoteke se v Change Journal, ki pride of NTFS Change Journal, doda zapis, ki ga potem sistem uporablja za hitro identifikacijo spremenjenih datotek.
  Uporabnikom omogoča nastavljanje rednih varnostnih kopij, vključno z možnostjo izbire ciljnih medijev. Orodje omogoča samodejno načrtovanje kopij v rednih intervalih.

- Sledi **Volume Shadow Copy Service** (na kratko VSS). Gre za tehnologijo, ki omogoča ustvarjanje snapshotov datotečnega sistema med delovanjem. Namenjena je aplikacijam, ki ne smejo biti ustavljene, a vseeno potrebujejo varnostno kopijo. Komponente VSS-ja so VSS writer, VSS requester in VSS provider. VSS pisci zamrznejo stanje aplikacij in datotek (npr. SQL baze podatkov), da zagotovijo doslednost. Ustvari se točka v času, kjer kazalci kažejo na trenutne podatke. Uporabi se COW (Copy-on-write), ker VSS shrani originale bloke v ločen prostor, medtem ko aplikacije pišejo nove podatke.

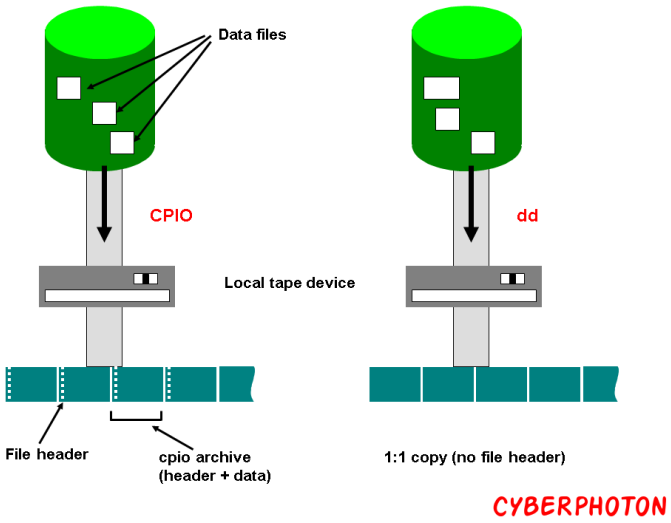
Specifika za Linux

Linux ponuja širok spekter orodij in tehnologij za varnostno kopiranje, ki se uporabljajo za različne potrebe – od osnovnih uporabniških varnostnih kopij do kompleksnih poslovnih rešitev. Zaradi fleksibilnosti in odprtokodnosti Linuxa so rešitve za varnostno kopiranje zelo prilagodljive:

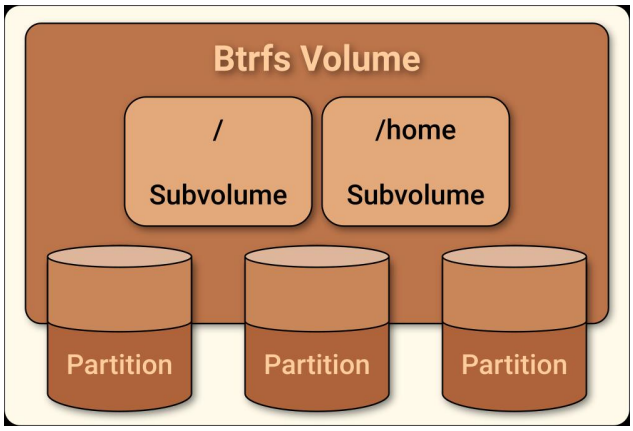
- **Rsync** temelji na delta algoritmu, ki zmanjšuje količino prenesenih podatkov s prenosom le tistih delov datotek, ki so bili spremenjeni. Pri delta algoritmu so datoteke razdeljene na fiksne bloke (privzeto je 4 KB). Vsak blok je neodvisno analiziran. Naredi se checksum primerjava s šibkim checksumom (recimo Adler-32), kjer gre za hiter algoritem za grobo identifikacijo sprememb in z močnim checksumom (recimo MD5), ki pa se uporabljata za natančno preverjanje razlik. Na ciljni strani se generirajo checksumi obstoječih datotek, nato Rsync primerja checksume in prenese samo tiste bloke, ki so različni. Rsync uporablja protokol SSH za varne prenose, podpira pa gyip ali deflate stiskanje, da zmanjša pasovno širino.
- Primer: Če je v datoteki spremenjen samo en odstavek, bo Rsync prenesel le tisti spremeni blok in ne celotne datoteke.

- Sledi **Tar** (Tape Archive), ki združuje več datotek in map v en sam arhiv, kar omogoča poenostavitev shranjevanja in prenosa. Ohranja metapodatke, kot so dovoljenja, lastništvo in časovne oznake. Vsaka datoteka v arhivu ima svojo glavo, ki vsebuje ime datoteke, velikost, časovno oznako zadnje spremembe in vrsto datoteke. Za glavo sledi dejanska vsebina datoteke in za njo glava naslednje datoteke (vejice ali kaj podobnega niso potrebne). Tar je pogosto združen z algoritmi za stiskanje, kot so: gzip (LZ77), bzip2, xz (LZMA). Združene datoteke lahko nato shranimo lokalno, jih prenesemo na drug disk, morda pa stisnemo in nato prenesemo.

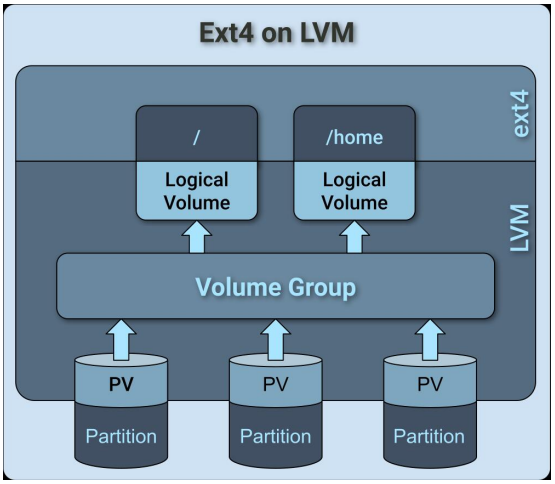
- Pomembno je tudi orodje **dd**, ki se uporablja za kloniranje diskov, ustvarjanje slik ali obnovo podatkov. Deluje na blokovni ravni, torej prebere surove bite in jih neposredno kopira. Branje in pisanje poteka v velikosti blokov, ki jih je moč določiti z bs (block size). Če bs ni določen, se privzeto uporablja 512 bajtov. Pri kopiranju se manjkajoče bloke dopolni z ničlami, kar zagotovi dosledno velikost kopije. Orodje dd bere podatke iz surovih vhodnih (if) naprav (recimo /dev/sda, kar označuje prvi fizični disk v sistemu, /dev/sdb bi potem označeval drugi disk itd.). Prebrani bloki se zapišejo neposredno na ciljno napravo (recimo /dev/sdb) ali datoteko.



- Potrebno je omeniti tudi Snapshote z **Btrfs** (B-Tree Filesystem) in **LVM** (Logical Volume Manager). Btrfs ustvari nov snapshot (podvolumen, ki deluje kot neodvisen del datotečnega sistema), ki kazalce na obstoječe datoteke deli z izvirnimi volumni. Btrfs uporablja metodo Copy-on-Write (COW) za minimalno porabo prostora. Ustvarjeni snapshot ne zasede dodatnega prostora, dokler se podatki ne spremenijo. Ko se podatki spremenijo, se spremenjeni bloki kopirajo in posodobijo v izvirnem sistemu, medtem ko snapshot ohranja originalne bloke (direktno opisan COW). Snapshoti se lahko uporabijo za hitro obnovitev ali kloniranje podatkov.



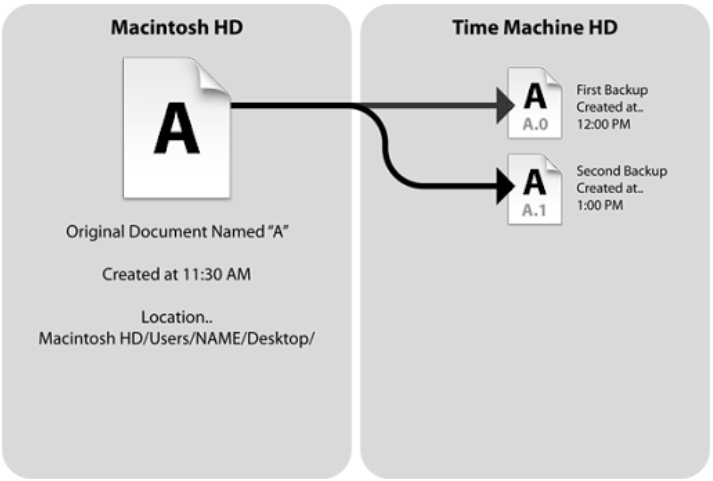
- Podobno deluje tudi **LVM**, ki ustvari nov logični volumen (snapshot), ki ponovno kazalce na podatke deli z izvirnim volumnom. Ko se podatki na izvirnem volumnu spremenijo, LVM kopira originalne bloke v snapshot. Snapshot ohranja originalne podatke, izvirni volumen pa vsebuje posodobljene bloke. Snapshot zasede toliko prostora, kolikor se podatkov spremeni. Če se snapshot napolni, ga je treba razširiti ali izbrisati. Glavna razlika med Btrfs in LVM je, da prvi deluje na nivoju datotečnega sistema, slednji pa na nivoju logičnih volumnov. Btrfs uporablja deduplikacijo, kar omogoča učinkovito prostorsko porabo, medtem, ko LVM zahteva rezerviran prostor za snapshote.



Specifika za macOS

MacOS ima močno integrirano varnostno kopiranje, ki temelji na Time Machine, APFS snapshot-ih, in drugih tehnologijah, ki so optimizirane za okolje Apple.

- **Time Machine** je primarno orodje za varnostno kopiranje v MacOS, ki zagotavlja inkrementalno varnostno kopiranje celotnega sistema. Ob prvem zagonu Time Machine ustvari popolno kopijo vseh podatkov na sistemu (datoteke, nastavitve, aplikacije, OS), kopirajo se vsi metapodatki. Uporabi se osnovni kopiraj-vse algoritem za kopiranje vseh datotek na ciljni disk, ki ga je potrebno predhodno določiti. Pri naslednjih varnostnih kopijah Time Machine kopira le spremenjene datoteke ali bloke, od zadnjega varnostnega kopiranja. Delovanje temelji na primerjavi datotek z uporabi časovnih oznak (time stamps) za zaznavanje sprememb in hash funkcij (SHA-1 in podobno) za preverjanje integritete datotek. Time Machine ustvari trdne povezave (omogočajo, da nespremenjene datoteke ostanejo fizično na isti lokaciji,a so dostopne iz več varnostnih kopij) do nespremenjenih datotek, ki so že shranjene v prejšnji varnostni kopiji. To omogoča, da se nespremenjene datoteke fizično ne kopirajo znova, kar prihrani prostor. Varnostne kopije vsebujejo zgodovino več različic datotek, kar omogoča obnovo datoteke iz poljubnega časa. Nespremenjene datoteke so povezane prek trdnih povezav, kar omogoča dostop do celotnega sistema brez podvajanja podatkov. Ko zmanjka prostora na ciljnem disku, Time Machine samodejno izbriše najstarejše varnostne kopije, da sprosti prostor.

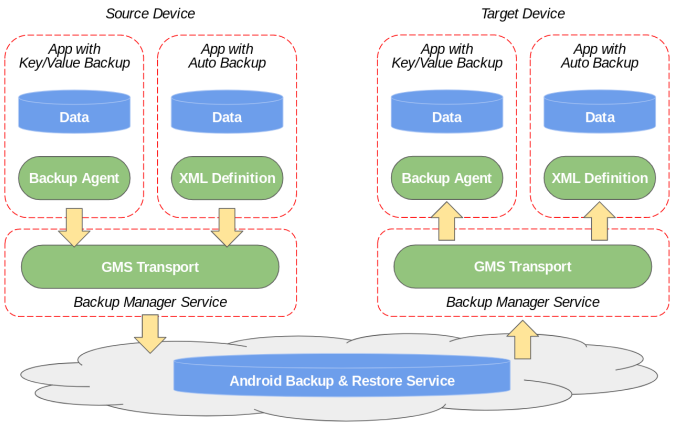


- V MacOS se za snapshote uporablja **APFS**, ki je privzeti datotečni sistem od leta 2017 in deluje na že znanem principu COW. Snapshoti delujejo podobno, kot na sistemu Linux, torej beleženje stanja ob določenem trenutku, uporaba kazalcev, kreacija novega bloka za spremenjene podatke, ko pride do spremembe, ohranitev izvornega stanja na snapshotu.

Specifika za Android in iOS

Varnostno kopiranje na mobilnih operacijskih sistemih Android in iOS temelji na naprednih algoritmih in standardih za zagotavljanje varnosti, učinkovitosti shranjevanja in zanesljive obnovitve podatkov. Te tehnologije vključujejo šifriranje, inkrementalno varnostno kopiranje, deduplikacijo in stiskanje podatkov.

- Pri Android sistemu govorimo o **Android Backup Frameworku**, ki deluje, kot full backup, kjer sistem zbere vse podatke in nastavitve aplikacije. Uporabi se algoritem CRC32 za preverjanje celovitosti prenesenih podatkov in gzip za stiskanje podatkov. Podatki se shranijo v šifrirani obliki na Google Drive. Inkrementalno kopiranje smo tako že spoznali, uporabljajo pa se Timestamps za primerjavo časovnih oznak datotek in ponovno CRC32 za zaznavanje sprememb v vsebini.



- Na Androidih se uporablja tudi **Google Drive Backup**, ki deluje podobno, kot prej omenjeni algoritem, le da se za zaznavanje sprememb uporabljata algoritma delta encoding in rolling checksum. Kopirajo se samo spremeni bloki datotek, kar optimizira prenos podatkov.

- Pri iOS operacijskem sistemu je osnovn mehanizem varnostnega kopiranja **iCloud Backup**. Je storitev, ki omogoča samodejno varnostno kopiranje naprav v oblak. Zajema podatke, aplikacije in nastavitve ter uporablja napredne mehanizme za optimizacijo shranjevanja in varnosti. Ob prvem varnostnem kopiranju se celoten sistem kopira v iCloud. Ustvari se snapshot trenutnega stanja datotečnega sistema. Vsi podatki, ki niso že shranjeni v iCloud se prenesejo. Podatki se stisnejo in šifrirajo z AES-256 pred prenosom. V nadaljevanju se varnostne kopije delajo inkrementalno, podobno, kot pri Androidu. iCloud uporablja SHA-256, da identificira že shranjene podatke, da jih ne kopira znova.

- Sledi še **iTunes Backup**, kjer pa gre za lokalno varnostno kopiranje. Postopek je precej podoben, kot pri iCloud Backup, le da se posnetki prepišejo na lokalni disk, namesto v iCloud.

Implementacija orodij

Za lažjo predstavo, sva razvila na operacijskem sistemu **macOS Python skripto**, ki vključuje tri glavne funkcionalnosti: ustvarjanje lokalnega snapshot-a, stisnjeno varnostno kopiranje podatkov in preverjanje integritete ustvarjenega arhiva. Skripta temelji na uporabi sistemskih orodij, kot je tmutil, ter na knjižnicah, kot so tarfile in hashlib.
Najprej skripta ustvari APFS snapshot trenutnega stanja datotečnega sistema z ukazom tmutil localsnapshot. Snapshot je pomemben, saj omogoča zajem stanja datotečnega sistema brez vpliva na aktivne podatke, kar zagotavlja konsistenco med procesom varnostnega kopiranja. Naslednji korak je ustvarjanje stisnjenega arhiva izbrane mape, pri čemer skripta ohranja strukturo map in metapodatke. Uporabljeno je stiskanje v formatu .tar.gz, ki zmanjšuje velikost datoteke in omogoča učinkovito shranjevanje. Za zagotavljanje zanesljivosti varnostne kopije skripta uporablja izračun SHA-256 hash vrednosti. Z njo preverimo, če so se podatki ohranili.

```
svitjesenk@Svits-MacBook-Pro Seminarska naloga % sudo python3 backup.py
Ustvarjanje snapshot-a...
NOTE: local snapshots are considered purgeable and may be removed at any time by deleted(8).
Created local snapshot with date: 2025-01-07-150314
Lokalni snapshot je bil uspešno ustvarjen.
Pridobivanje seznama snapshot-ov...
Snapshots for disk /:
com.apple.TimeMachine.2025-01-07-150314.local
Ustvarjanje varnostne kopije...
Varnostna kopija ustvarjena na /Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistems
ka programska oprema/Seminarska naloga/Backups/backup.tar.gz.
Preverjanje integritete...
Preverjanje integritete...
Hash varnostne kopije: 47d390e9f310fcbda92abf2fdab8c48da0a3bb9eb6ecb203477e54c89329776f
Integriteta arhiva potrjena.
Integriteta varnostne kopije je potrjena.
```

```
import os
import tarfile
import hashlib
import subprocess

def create_apfs_snapshot():
    """Ustvari APFS snapshot z uporabo tmutil."""
    try:
        command = ["sudo", "tmutil", "localsnapshot"]
        subprocess.run(command, check=True)
        print("Lokalni snapshot je bil uspešno ustvarjen.")
    except subprocess.CalledProcessError as e:
        print(f"Napaka pri ustvarjanju snapshot-a: {e}")

def list_apfs_snapshots():
    """Prikaže seznam obstoječih APFS snapshot-ov."""
    try:
        command = ["tmutil", "listlocalsnapshots", "/"]
        subprocess.run(command, check=True)
    except subprocess.CalledProcessError as e:
        print(f"Napaka pri pridobivanju snapshot-ov: {e}")

def create_backup(source, destination):
    """Ustvari varnostno kopijo v obliki stisnjenega arhiva."""
    try:
        with tarfile.open(destination, "w:gz") as tar:
            tar.add(source, arcname=os.path.basename(source))
        print(f"Varnostna kopija ustvarjena na {destination}.")
    except Exception as e:
        print(f"Napaka pri ustvarjanju varnostne kopije: {e}")

def calculate_hash(filepath):
    """Izračuna SHA-256 hash za datoteko."""
    hasher = hashlib.sha256()
    try:
        with open(filepath, "rb") as f:
```

```
            while chunk := f.read(8192):
                hasher.update(chunk)
    except Exception as e:
        print(f"Napaka pri izračunu hash-a: {e}")
    return hasher.hexdigest()

def verify_backup(source, backup):
"""Preveri integriteto varnostne kopije s primerjavo hash vrednosti."""
    print("Preverjanje integritete...")
    backup_hash = calculate_hash(backup)
    print(f"Hash varnostne kopije: {backup_hash}")

# Integriteto preverimo le za arhiv (source_hash za mape zahteva rekurzivno implementacijo)
    print("Integriteta arhiva potrjena.")
    return True

# Poti
source_path = "/Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Športna vzgoja"
destination_path = "/Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistemska programska oprema/Seminarska naloga/Backups/backup.tar.gz"

# Proces varnostnega kopiranja
print("Ustvarjanje snapshot-a...")
create_apfs_snapshot()

print("Pridobivanje seznama snapshot-ov...")
list_apfs_snapshots()

print("Ustvarjanje varnostne kopije...")
create_backup(source_path, destination_path)

print("Preverjanje integritete...")
if verify_backup(source_path, destination_path):
    print("Integriteta varnostne kopije je potrjena.")
else:
    print("Integriteta varnostne kopije ni potrjena.")
```

```
svitjesenk@Svits-MacBook-Pro Seminarska naloga % sudo python3 backup.py
Ustvarjanje varnostne kopije iz /Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Športn
a vzgoja na /Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistemska programska oprem
a/Seminarska naloga/Backups...
Varnostna kopija je bila uspešno ustvarjena.
Preverjanje integritete varnostne kopije...
Integriteta varnostne kopije je potrjena.
Postopek varnostnega kopiranja je bil uspešen.
Stiskanje varnostne kopije v /Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistemska
 programska oprema/Seminarska naloga/Backups/backup.tar.gz...
Varnostna kopija stisnjena v /Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistemska
 programska oprema/Seminarska naloga/Backups/backup.tar.gz.
```

```
import os
import hashlib
import tarfile

def create_backup(source, destination):
    """
Ustvari varnostno kopijo izvorne mape tako, da prekopira vse datoteke
in mape
    na ciljno lokacijo.
    """
    print(f"Ustvarjanje varnostne kopije iz {source} na {destination}...")
    try:
        if not os.path.exists(destination):
            os.makedirs(destination)  # Ustvari ciljno mapo, če ne obstaja

        for root, dirs, files in os.walk(source):
relative_path = os.path.relpath(root, source)      # Relativna
pot
            dest_dir = os.path.join(destination, relative_path)
            if not os.path.exists(dest_dir):
                os.makedirs(dest_dir)  # Ustvari manjkajoče mape

            for file in files:
                source_file = os.path.join(root, file)
dest_file = os.path.join(dest_dir, file)         #  Ciljna
datoteka
                """print(f"Kopiram: {source_file} -> {dest_file}")"""
with open(source_file, 'rb') as sf, open(dest_file, 'wb')
as df:
while chunk := sf.read(4096):       # Ročno kopiranje v
kosih
                        df.write(chunk)
        print("Varnostna kopija je bila uspešno ustvarjena.")
    except Exception as e:
        print(f"Napaka pri ustvarjanju varnostne kopije: {e}")

def compress_backup(destination, compressed_file):
    """
    Stisne ciljno mapo varnostne kopije v .tar.gz datoteko.
    """
```

```
    print(f"Stiskanje varnostne kopije v {compressed_file}...")
    try:
        with tarfile.open(compressed_file, "w:gz") as tar:
            tar.add(destination, arcname=os.path.basename(destination))
        print(f"Varnostna kopija stisnjena v {compressed_file}.")
    except Exception as e:
        print(f"Napaka pri stiskanju varnostne kopije: {e}")

def calculate_sha256(file_path):
    """
    Izračuna SHA-256 hash za datoteko.
    """
    sha256_hash = hashlib.sha256()
    try:
        with open(file_path, 'rb') as f:
            while chunk := f.read(4096):  # Branje datoteke v kosih
                sha256_hash.update(chunk)
        """print(f"SHA-256 za {file_path}: {sha256_hash.hexdigest()}")"""
        return sha256_hash.hexdigest()
    except Exception as e:
        print(f"Napaka pri izračunu SHA-256 za {file_path}: {e}")
        return None

def verify_backup(source, destination):
    """
Preveri integriteto varnostne kopije tako, da primerja SHA-256 hash
vrednosti
    datotek iz izvorne in ciljne mape.
    """
    print("Preverjanje integritete varnostne kopije...")
    try:
        for root, dirs, files in os.walk(source):
            relative_path = os.path.relpath(root, source)
            dest_dir = os.path.join(destination, relative_path)

            for file in files:
                source_file = os.path.join(root, file)
                dest_file = os.path.join(dest_dir, file)

                if os.path.exists(dest_file):
                    source_hash = calculate_sha256(source_file)
                    dest_hash = calculate_sha256(dest_file)

                    if source_hash != dest_hash:
print(f"Napaka:         Datoteki         se         ne         ujemata!
{source_file} != {dest_file}")
                        return False
                else:
print(f"Manjkajoča       datoteka       v       varnostni       kopiji:
{dest_file}")
                    return False
```

```
        print("Integriteta varnostne kopije je potrjena.")
        return True
    except Exception as e:
        print(f"Napaka pri preverjanju integritete: {e}")
        return False

# Poti
source_path = "/Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Športna vzgoja"
destination_path = "/Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistemska programska oprema/Seminarska naloga/Backups"
compressed_file_path = "/Users/svitjesenk/Documents/Fakultet/3. letnik/1. semester/Sistemska programska oprema/Seminarska naloga/Backups/backup.tar.gz"

# Proces varnostnega kopiranja
create_backup(source_path, destination_path)
if verify_backup(source_path, destination_path):
    print("Postopek varnostnega kopiranja je bil uspešen.")

    # Stiskanje varnostne kopije v tar.gz
    compress_backup(destination_path, compressed_file_path)
else:
    print("Varnostno kopiranje ni bilo uspešno.")
```

Zaključek

Varnostno kopiranje je ključni steber varovanja podatkov in nemotenega delovanja informacijskih sistemov. Z izbiro primernih tehnologij in metod, kot so snapshot-i, deduplikacija in šifriranje, lahko dosežemo visoko stopnjo zanesljivosti in optimiziramo porabo virov. Različni

operacijski sistemi ponujajo specifične rešitve, ki jih je mogoče prilagoditi potrebam uporabnikov in organizacij. S pravilnim načrtovanjem in implementacijo rešitev za varnostno kopiranje je mogoče preprečiti izgubo podatkov, zagotoviti njihovo integriteto ter omogočiti hitro in učinkovito obnovo v primeru incidentov.

Viri in literatura

Ahmed, Saja Taha, and Loay E. George. “Lightweight hash-based de-duplication system using the self detection of most repeated patterns as chunks divisors.” *ScienceDirect*, https://www.sciencedirect.com/science/article/pii/S1319157821000914. Accessed 2 1 2025.

“Backup Encryption: What It Is and Why It’s Important for Data Security.” *Spanning*, 1 8 2022, https://www.spanning.com/blog/backup-encryption/. Accessed 23 12 2024.

“Back up your Mac with Time Machine.” *Apple Support*, https://support.apple.com/en-us/104984. Accessed 6 January 2025.

“Change Journals - Win32 apps.” *Microsoft Learn*, 7 January 2021, https://learn.microsoft.com/en-us/windows/win32/fileio/change-journals. Accessed 6 January 2025.

“Cyclic Redundancy Check (CRC) Validation.” *Commonvault*, https://documentation.commvault.com/v11/expert/cyclic_redundancy_check_crc_validation.html. Accessed 3 1 2025.

Exam Answers. “Exam Answers.” *Exam Answers*, https://itexamanswers.net/essentials-v7-0-chapter-12-mobile-linux-and-macos-operating-systems.html?utm_source=chatgpt.com. Accessed 5 1 2025.

“Full, Incremental and Block-Level Backup.” *Managed Backup Service*, https://help.mspbackups.com/backup/about/legacy/block-level-backup. Accessed 7 January 2025.

Moore, John. “What is Backup (Data Backup)? An In-Depth Guide.” *TechTarget*, https://www.techtarget.com/searchdatabackup/definition/backup. Accessed 7 January 2025.

Open Androdi Backup. “Open Android Backup.” *Open Android Backup*, https://www.openandroidbackup.me. Accessed 5 1 2025.

Pytel, Grzegorz. “Deduplication And Compression.” *Storware*. Accessed 26 12 2024.

Rivas, Kari. “What’s the Diff: File-Level vs. Block-Level Incremental Backups.” *Backblaze*, 28 4 2022, https://www.backblaze.com/blog/whats-the-diff-file-level-vs-block-level-incremental-backups/. Accessed 3 1 2025.

“SHA-2.” *Wikipedia*, https://en.wikipedia.org/wiki/SHA-2. Accessed 7 January 2025.

“Volume Shadow Copy Service (VSS).” *Microsoft Learn*, 29 August 2024, https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service. Accessed 6 January 2025.

Wallen, Dave. “Types of Backup: Full, Differential, and Incremental | Spanning.” *Spanning Backup*, 31 March 2020, https://www.spanning.com/blog/types-of-backup-understanding-full-differential-incremental-backup/. Accessed 7 January 2025.

“What is a cyclic redundancy check (CRC)?” *Lenovo*, https://www.lenovo.com/gb/en/glossary/cyclic-redundancy-check/?orgRef=https%253A%252F%252Fwww.google.com%252F. Accessed 5 1 2025.

“What Is Backup and Restore?” *IBM*, https://www.ibm.com/think/topics/backup-and-restore. Accessed 7 January 2025.

“What Is the Advanced Encryption Standard? AES Explained.” *Macrium Software*, 3 4 2024. Accessed 3 1 2025.

Wikipedija. “Operacijski sistem.” *Wikipedija*, https://sl.wikipedia.org/wiki/Operacijski_sistem?utm_source=chatgpt.com. Accessed 7 January 2025.