Електронски факултет

Александра Медведева 14, Ниш

Семинарски рад

Форензика фајл система

Професор:

проф. др. Братислав Префић

Студент:

дипл. инг. Милош Лазовић, 1715

Ниш, Јун 2024. године

Садржај

1. Увод у форензику фајл система

У данашњем дигиталном друштву, где се скоро сваки аспект нашег живота одвија путем дигиталних уређаја и система, анализа дигиталних трагова постаје кључна. У том контексту, форензичка анализа фајл система игра виталну улогу у откривању, анализи и интерпретацији дигиталних доказа који су од суштинског значаја за истражне поступке, правне случајеве и сигурносне инциденте. Форензичка анализа фајл система пружа алате, технике и методологије за дубинску анализу података унутар фајл система, омогућавајући стручњацима да разумеју на који начин су подаци креирани, модификовани, брисани или премештани током времена. Ова дисциплина игра кључну улогу у многим областима, укључујући кривично право, безбедност информација, дигитално истраживање, администрацију система и управљање инцидентима. Анализа фајл система омогућава идентификацију неправилности, необичних активности и неовлашћеног приступа подацима, што доприноси откривању претњи, истраживању злочина, доказивању кривице или невиности, као и унапређењу сигурности и заштите дигиталних система. У том смислу, дубоко разумевање и ефикасна примена форензичких техника и алата постају од суштинског значаја за решавање комплексних ситуација и случајева који захтевају анализу дигиталних трагова.

2. Организација и принцип рада хард-диска

Хард-дискови су се појавили на технолошкој сцени 1956. године, као изум компаније IBM, и након свог изласка на тржиште постали су доминантна технологија за секундарно складиштење података у типичним рачунарским системима током 1960-их. Данас се хард-дискови користе у многим рачунарским системима. Недавне иновације су технологије попут хард-дискова без покретних делова које користе флеш меморију и NAND технологије, полако почињу да потискују магнетске дискове с појединих система као што су преносници. Хард-дискови због непрекидног развоја и усвајања нових технолошких решења још су најпривлачнији као технологија за секундарно складиштење података што се тиче односа следећих својстава: носивости, брзине преноса података и цене.

Слика 2.1. – Изглед хард-диска без кућишта

Хард-дискови састоје се од једног или више кружних плоча затворених у херметичком кућишту. Ове кружне плоче врте се око једне средишње осе уз помоћу електромотора. Плоче су обично направљене од легуре неког метала (гвожђа, алуминијума или неке друге комбинације), док су код неких хард-дискова плоче израђене од стакла. У процесу израде плоче се обично пресвлаче танким слојем неке феромагнетске материје. Читање и записивање података врши се уз помоћ посебне магнетске главе која лебди непосредно изнад магнетског слоја. Код хард-дискова с више плоча, магнетске главе су на свакој плочи и некада с обе стране плоче. Магнетске главе постављају се на посебне носаче који је управљан с посебним механизмом и управљачком јединицом. Подаци који се записују односно читају преко магнетских

глава користе разне кодне и модулацијске системе (FM, MFM, MMFM). Предност хард-дискова је да се подаци могу читати:

- у следу, један за другим, или
- ван реда с било које тачке коју магнетска глава може досећи.

Ова особност приступа ван реда, довела је до револуције у обради података на рачунарским системима, јер више није било потребно читати магнетске траке од почетка до места где се налазе подаци, или читати бушене картице које су заузимале много простора и на које није било могуће спремити много података. Ова иновација омогућила је развој нових операционих система, као и развој нових апликацијских програма у којима се подаци могу обрадити у стварном времену.

2.1. Организација података

Подаци се снимају на једну или обе површине сваке плоче (диска), у концентричним круговима. Један такав круг (на једној површини) се назива стаза, траг или трака. Свакој стази се додељује адреса. Уколико имамо 10000 стаза на плочи, стазе се адресирају од 0 до 9999, од спољашње ка унутрањим, респективно.

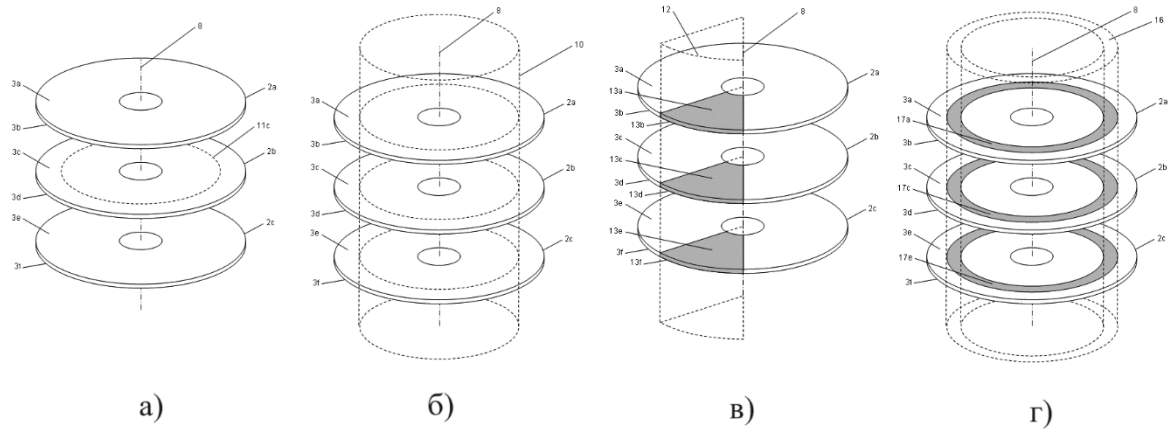
Скуп свих стаза једнаке удаљености од центра ротације (па тим и пречника), односно свих стаза на истој адреси, назива се „цилиндар“. Цилиндар 0 садржи све стазе са адресом 0, на свакој плочи на хард-диску.

Подаци нису континуални већ је свака стаза угаоно подељена у више блокова који се називају „секторима“. Уобичајена дужина сектора је 512 бајтова, не рачунајући додатне податке за позиционирање и контролу и корекцију грешака. Сектори представљају најмању адресибилну јединицу на хард-диску, и сваком сектору се додељује адреса почевши од 1, за сваку стазу.

Ради релативног одржања густине података преко целе површине дискова, спољни цилиндри могу имати више сектора него унутрашњи. Скуп суседних цилиндара са једнаким бројем сектора се назива „зона“. Коначна адреса сектора се добија тако што узимамо:

- адресу цилиндра како бисмо добили стазу на којој се налази сектор,
- адресу матнетне главе како бисмо добили плочу и страну на којој се налази сектор, и
- адресу сектора на стази

Овај начин адресирања се зове цилиндар-глава-сектор (CHS, cylider-head-sector) и не користи се као главна метода адресирања података на диску. Уместо ње користи се метода која се заснива на адресама логичких блокова, LBA (Logical Block Address).



Слика 2.2. – Организација података на диску, а) стаза или трака, б) цилиндар, ц) сектор, г) зона

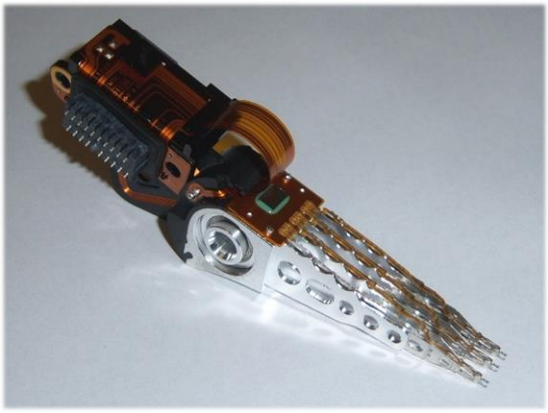
На диску се налази покретна рука која на свом крају садржи матнетну главу. Постоји по једна магнетна глава за читање са обе стране плоче. Свака глава се такође адресира како бисмо знали на којој плочи на на којој страни плоче се обавља читање или упис.

Сектор може постати неисправан и такав сектор не би требало више користити за складиштење корисничких података. Код старијих дискова, одговорност оперативног система је била да зна где се налазе лоши сектори и да такве секторе не користи за смештање датотека. Корисници су такође могли ручно да кажу диску које секторе да игнорише јер су неисправни. Многи фајл системи и даље пружају опцију за означавање сектора као неисправних. Међутим, то обично није потребно, јер модерни дискови могу идентификовати неисправан сектор и премапирају адресу на добру локацију негде другде на диску. Цео тај процес се обавља без знања корисника.

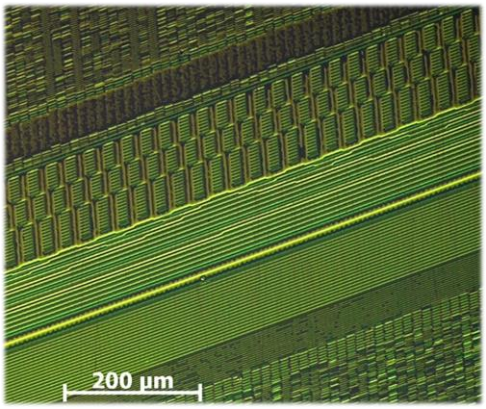
2.2. Складиштење и читање података

Подаци се на диск уписују уз помоћ мале спирале која је саставни део главе. Спирала у бираним тренуцима пропушта електричну струју изабраног смера. Магнетска глава састоји се од спирале која је намотана на тврдо феритно језгро. Глава је учвршћена на ручицу коју по диску помера покретач. Уз помоћ њега, глава може да се помера над целим полупречником диска. Магнетска површина плоче у диску је подељена на пуно малих магнетских подручја величине микрометра, а свака од тих површина се користи за чување (кодирање) једног бита информације. До 2005. та подела магнетске површине је била само хоризонтална, али од тада до данас та подела је и вертикална, чиме су добијени хард-дискови већег капацитета (до 2 TB). Због природне кристалне структуре магнетских материјала, те регије на диску се састоје од неколико стотина магнетских честица (једна магнетска честица је величине 10 nm). Протицањем струје кроз спиралу ствара се магнетско поље које се због близине главе протеже и кроз магнетски материјал на површини диска. Како се диск брзо окреће испод главе, сав материјал који прође испод главе се магнетизује у смеру одређеном смером протицања електричне струје. Укључивањем струје у краткотрајним бинарним тренуцима, постиже се на површини диска низ различито магнетизованих подручја једно иза

другог, чиме је на диск записан низ података, тј. битова. Подаци се на диску налазе као низ магнетских честица на магнетском слоју диска које су смештене у концентричне кругове.



Слика 2.3. – Магнетске главе за записивање и читање података (https://upload.wikimedia.org/wikipedia/commons/5/54/Kopftraeger_WD2500JS-00MHB0.jpg)



Слика 2.4. – Изглед магнетног записа на диску (https://upload.wikimedia.org/wikipedia/commons/c/ca/Aufnahme_einzelner_Magnetisierungen_gespeicherter_Bits_auf_einem_Festplatten-Platter..jpg)

Читање се на почетку радило користећи чињеницу да када низ различито магнетизованих подручја брзо прође испод спирале магнетске главе, у спирали се индукује електрични напон код сваке промене поља. Индуковани напон и тако добијена струја имају своју јачину, која зависи од јачине магнетског поља, његовог смера, брзине промене магнетског поља испред главе и удаљености главе од диска. Због разлике у индукованом напону на спирали у одређеном тренутку добија се напонски сигнал. Из тог напонског сигнала се може закључити какав је распоред магнетизованих подручја прошао испод ње и тиме се низ битова прочитао. Данас се користе друге магнетске појаве, рецимо особине да присутност магнетског поља мења електричну отпорност неког материјала. Код таквих дискова, глава је магнетоотпорна. Приликом проласка читајуће главе преко магнетизоване површине диска, читајућа глава мења свој електрични отпор због промене јачине и смера магнетског поља.

У данашњим хард-дисковима главе за читање и писање су одвојене, за разлику од старих дискова на којима се све обављало уз помоћ једне главе. Читајућа глава је магнетоотпорна, док је пишућа глава танкослојна и индуктивна.

Добра својства магнетског диска јесу велики капацитет, постојаност података и брзи приступ подацима. Негативна својства јесу:

- осетљивост на прљавштину,
- електромагнетска поља, и
- ограничење максималне густине података.

Магнетски диск је посебно осетљив на електромагнетска поља и при руковању треба имати то на уму.

2.3. Типови адреса сектора

Физичка адреса сектора је адреса која је релативна на почетак физичког медијума. Постоје две различите методе за адресирање физичких адреса. Метода која се користи код старијих хард-дискови се заснива на физичку орханизацију самог диска, и то је цилиндар-глава-сектор – CHS (Cylinder-head-sector) метода. Код новијих дискова корити се метода која се заснива на адресама логичких блокова, LBA (Logical Block Address).

CHS систем уводи одређена ограничења због којих се данас ретко користи. Оригинална ATA спецификација користила је 16-битну вредност цилиндра, 4-битну вредност главе и 8-битну вредност сектора, али су старији BIOS-и користили 10-битну вредност цилиндра, 8-битну вредност главе и 6-битну вредност сектора. Дакле, да би комуницирао са хард диском преко BIOS-а, морала је бити коришћена најмања величина за сваку вредност, што је дозвољавало само диск од 504 MB.

Да би се превазишао овај проблем и ограничење од 504 MB, нови BIOS-и су развијани тако да преводе опсеге адреса који они корсите у опсеге адреса који су одређени ATA спецификацијом. На пример, уколико апликација захтева податке са цилидра 8, главе 4 и сектора 32, BIOS би ту адресу превео у адресу сектора на цилиндру 26, главе 2 и сектора 32. Овај процес превођења не ради за дискове који су већи од 8.1 GB. Ограничење од 8 GB потиче из начина на који су старији BIOS-и и хардверске спецификације адресирале простор на хард дисковима користећи CHS методу. Овај метод користио је ограничени број битова за представљање вредности цилиндра, главе и сектора, што је резултирало ограничењем у укупној адресабилној просторној величини диска. На пример, старији BIOS-и су користили 24-битни систем адресирања, где је максимална адреса била 2^24 сектора, што је еквивалентно 8 GB простора (јер је сваки сектор обично био величине 512 бајтова).

Да би се превазишао овај лимит прешло се на LBA методу адресирања. LBA адреса 0 одговара CHS адреси 0,0,1. LBA адреса 1 одговара CHS адреси 0,0,2. Када су сви сектори из стазе искоришћени прелази се на следећу главу на истом цилиндру, односно на адресу 0,1,1.

Метода за конверзију CHS у LBA:

$$LBA = \Big(\big((CYLINDER * HeadsPerCylinder) + HEAD\big) * SectorsPerTrack\Big) + SECTOR - 1$$

CYLINDER, HEAD, SECTOR, представљају делове CHS адресе.

Уколико имамо диск који има 16 глава по цилиндру и 63 сектора по стази, CHS adresa 2,3,4 би се превела у LBA на следећи начин:

$$\Big(\big((2 * 16) + 3\big) * 63\Big) + 4 - 1 = 2208$$

3. Анализа волумена

Анализа волумена укључује испитивање структура података које су повезане са партиционисањем и састављањем скупа бајтова како бисмо добили волумене, на уређајима за складиштење.

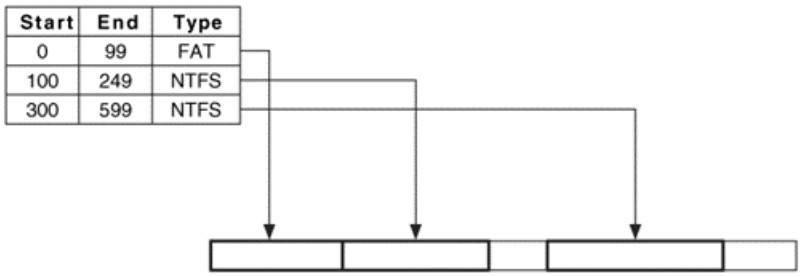
Термини партиција и волумен, се често користе заједно или један уместо другог, па их је због тога потребно јасно дефинисати.

Волумен представља колекцију адресибилних сектора које оперативни систем или апликација може да користи за складиштење података. Сектори у волумену не морају да буду на узастопним меморијским локацијама на диску, довољно је само да дају утисак као да јесу. Волумен може да буде и резултат спајања више мањих волумена. Уколико посматрамо цео хард-диск, он би представљао волумен код кога су сви сектори узастопни.

Партиција представља колекцију узастопних сектора унутар једног волумена. По дефиницији, свака партиција уједно представља и волумен.

Различити оперативни системи и хардверске платформе користе различите методе за партиционисање. Обично се користе једна или више табела, где свака ставка у табели описује једну партицију. Свака ставка од података садржи:

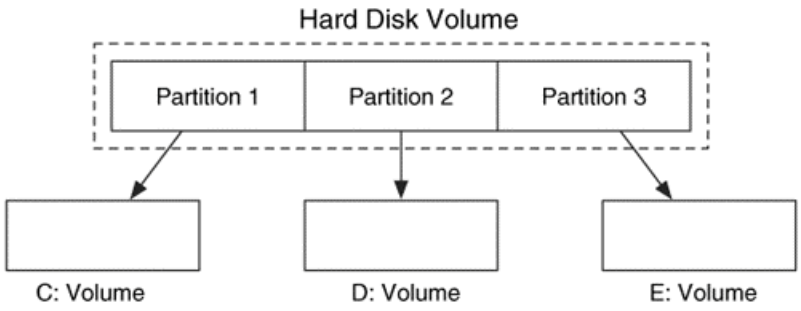
- почетни сектор партиције,
- крајњи сектор партиције или дужину партиције и
- тип партиције.



Слика 3.1. – Табела са основним подацима о свакој партицији
(File System Forensic Analysis, Brian Carrier – Figure 4.2.)

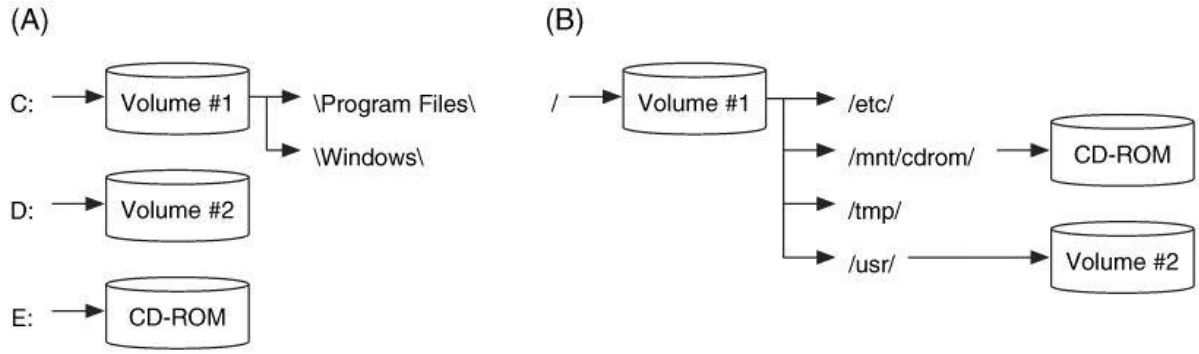
Сврха система за партиционисање је да организује распоред волумена. Најбитнији подаци су почетна и крајња локација сваке партиције. Разлог је и то што се први и послењи сектор је означавају ни на који посебан начин како би се назначиле границе партиције.

Код Windows оперативног система волумен хард диска може да се подели на више мањих партиција, којима се додељује слово за идентификацију. На слици је хард-диск подељен на три партиције, C:, D: и E:.



Слика 3.2. – Организација хард диска у три партиције
(File System Forensic Analysis, Brian Carrier – Figure 4.1.)

Код Unix оперативних система партиције се не представљају као засебни дискови C:, D: і E: већ као директоријуми који почиљу од корена директоријума "/". Поддиректоријуми директоријума "/" су или поддиректоријуми у истом фајл систему, или су тачке приступа (mounting point) за нове фајл системе и волумене. На пример, CD-ROM би добио ознаку диска E: у Windows-у, а у Linux-у може бити приказан у облику директоријума „/mnt/cdrom". Ово омогућава кориснику да променом директоријума мења дискове, и у многим случајевима корисник није свестан да је то учинио. Да би се минимизовао утицај оштећења дискова и побољшала ефикасност, Unix обично партиционише сваки диск у неколико волумена. Волумен за коренски директоријум „/" садржи основне информације, одвојени волумен може постојати за корисничке директоријуме „/home/", а апликације се могу налазити у сопственом волумену „/usr/".



Слика 3.3. – Приступне тачке два волумена и CD-ROM-а, код Windows система – а), и код UNIX система – б)
(File System Forensic Analysis, Brian Carrier – Figure 4.3.)

Најчешћа метода за адресирање сектора је LBA метода, као што је претходно поменуто. Код ове методе први сектор се адресира вредношћу 0, и та адреса представља физичку адресу сектора.

Пошто волумен представља колекцију сектора потребно је да обавимо адресирање сектора и унутар волумена. Логичка адреса волумена је адреса сектора у том волумену која је релативна у односу на почетак тог волумена. Уколико цео хард-диск посматрамо као један волумен, тада је физичка адреса хард-диска једнака логичкој

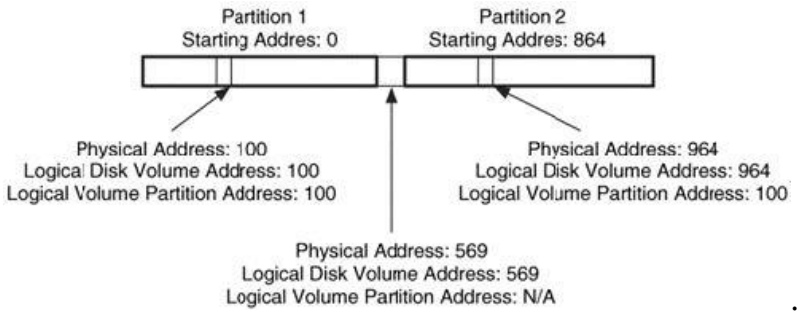
адреси волумена. Почетне и крајње адресе партиција су обично одређене логичким адресама волумена.

Када причамо о садржају партиције, он оним секторима који су дедељени конкретној партицији, онда имамо још један ниво логичких адреса волумена. Ове адресе су релативне у односу на почетак и крај партиције, а не на почетак диска или родитељског волумена. Ове адресе разликујемо тако што испред речи волумен убацујемо „диск“ или „партиција“:

- логичка адреса партиције волумена
- логичка адреса диска волумена

Уколико сектор није додељен партицији, онда он неће имати логичку адресу партиције волумена.

Код примера на слици прва партиција почиње у сектору 0, тако да су логичне адресе партиције волумена у њој исте као и логичне адресе диска волумена. Друга партиција почиње на физичком сектору 864 и логичне адресе диска волумена ових сектора су веће за 864 сектора од њихових логичних адреса партиције волумена.



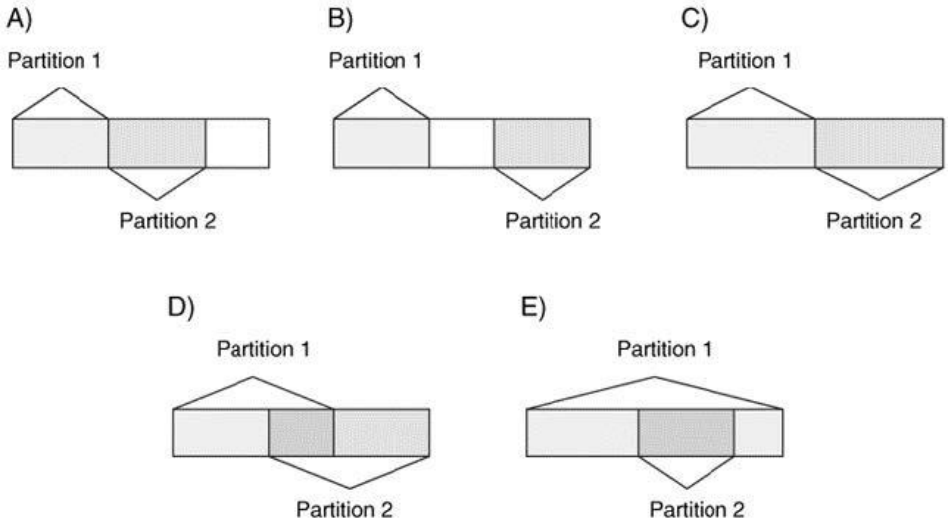
Слика 3.4. – Примери сектора са приказаним логичким адресама партиције волумена и логичким адресама диска волумена.
(File System Forensic Analysis, Brian Carrier – Figure 4.5.)

Основна теорија код анализе волумена је једноставна, за системе партиција желимо да локализујемо табеле партиција и обрадимо их како бисмо идентификовали њихов међусобни распоред и локације. Информације о распореду се затим прослеђују алату за анализу фајл система који треба да зна померај партиције, или се ти подаци исписују кориснику тако да он сам може одредити који подаци треба да буду анализирани.

При анализи система волумена, корисно је проверити сваку партицију у односу на остале партиције. Ово може послужити као једна од основних провера пре било какве даље анализе како би се утврдило где се још могу налазити тражени подаци или докази, осим у свакој познатих партицији. Већина система за партиционисање не захтева да ставке у табелама буду у сортираном редоследу, па је потребно да или корисник или алат за анализу обави сортирање на основу почетне и завршне локације пре него што се обави провера конзистентности.

Код прва провере испитује се крајња адреса последње партиције (адреса последњег сектора у тој партицији) са крајњом адресом родитељског волумена те партиције. Идеално, крајња адреса последње паритиције би требала да буде једнака адреси последњег сектора у том волумену.

Код следеће провере потребно је упоредити почену и крајњу адресу суседних партиција, у тим ситуацијама имамо четири различита сценарија.



Слика 3.5. – Примери како две партиције могу међусобно да буду организоване
(File System Forensic Analysis, Brian Carrier – Figure 4.6.)

У првом сценарију последња партиција се завршава пре краја волумена, што значи да могу да постоје делови диска који би могли садржати скривене или обрисане податке.

Други сценаријо, приказан на примеру B), је валидан, и код њега постоје сектори између две партиције који нису у партицији. Сектори који нису у партицији могли би бити коришћени за сакривање података и треба их анализирати.

Трећи сценаријо, приказан на примеру C), је оно што скоро сваки систем има, а то је да друга партиција почиње одмах након прве.

Четврти сценаријо, приказан на примеру D), обично је невалидан. Код њега друга партиција почиње пре него што прва партиција заврши. Ово ствара преклапање, и у многим случајевима ово је показатељ да је табела партиција оштећена. Да се утврдило која је партиција валидна, уколико таква постоји, потребно је анализирати податке сваке паритције.

Четврти сценаријо приказан на примеру E:, такође је обично неисправан. Друга партиција је унутар прве партиције, садржаје сваке партиције треба анализирати да бисте утврдили где је грешка.

4. Анализа фајл система

Анализа фајл система укључује неколико корака, почевши од идентификације релевантних фајл система на медијуму, преко екстракције метаподатака, до реконструкције и анализе садржаја фајлова. Сваки од ових корака захтева пажљиво планирање и извођење како би се осигурала тачност и интегритет доказа. Овим поступком анализе испитују се податци на волумену (тј. на партицији или диску) и интерпретирају их као фајл систем. Постоји много крајњих резултата овог процеса, али примери укључују листање фајлова у директоријуму, опоравак обрисаног садржаја и преглед садржаја сектора.

Фајл системи имају специфичне процедуре и структуре које се могу користити за складиштење једног фајла на флопи-диску или десетина хиљада фајлова на хард-диску. Свака инстанца фајл система има јединствену величину, али њена основна структура омогућава сваком рачунару који подржава ту врсту фајл система да је обради. Неки подаци захтевају унутрашњу структуру и организацију унутар фајла. Ово је слично као што физички документи захтевају структуру у облику одељака и поглавља.

Корисно је имати основни референтни модел како би се различити фајл системи лакше упоређивали. Поседовање таквог референтног модела такође олакшава одређивање где се докази могу налазити. На пример, референтни модел олакшава поређење разлика између FAT и Ext3 фајл система. За овај основни модел користићемо пет категорија:

1. фајл систем,
2. садржај датотека,
3. метаподаци,
4. назив фајла и
5. апликација.

Сви подаци у фајл систему припадају једној од ових категорија на основу улоге коју играју у фајл систему.

Категорија фајл система садржи опште информације о фајл систему. Сви фајл системи имају општу структуру, али свака инстанца фајл система је јединствена јер има јединствену величину и може бити модификована како би обезбедила боље перформансе. Подаци у категорији фајл система могу вам рећи где да пронађете одређене структуре података и колика је величина јединице података. Подаци у овој категорији могу да се посматрају као мапа за неки одређени фајл систем.

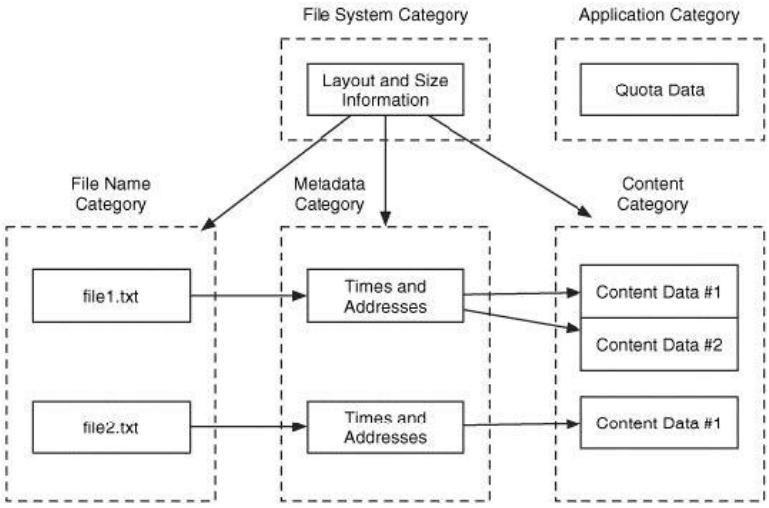
Категорија садржаја датотека садржи податке који чине стварни садржај онога што се налази у датотекама, што је и разлог постојања фајл система. Већина података у фајл систему припада овој категорији, а обично је организована у збирку контејнера. Сваки фајл систем додељује различита имена овим контејнерима, као што су кластери и блокови, а као општи термин може да се користи и *јединица података*.

Категорија метаподатака садржи податке који описују фајл; то су подаци који описују друге податке. Ова категорија укључује информације као што су место где је садржај фајла смештен, колика је величина фајла, времена и датуми када је фајл последњи пут

читан или писан, и информације о контроли приступа. Важно је напоменути да ова категорија не садржи садржај фајла и може не садржи ни име фајла. Примери структура података у овој категорији укључују:

- ставке FAT директоријума,
- ставке NTFS Master File Table,
- inode структуре UFS и Ext3 система

Категорија назива фајла, или категорија корисничког интерфејса, садржи податке који додељују име сваком фајлу. У већини фајл система, ови подаци се налазе у садржају директоријума и представљају листу имена фајлова са одговарајућом адресом метаподатака. Категорија назива фајла је слична имену хоста у мрежи. Мрежни уређаји комуницирају међусобно користећи IP адресе, које су тешке за памћење људима. Када корисник унесе име хоста удаљеног рачунара, локални рачунар мора да преведе то име у IP адресу пре него што комуникација може да започне.



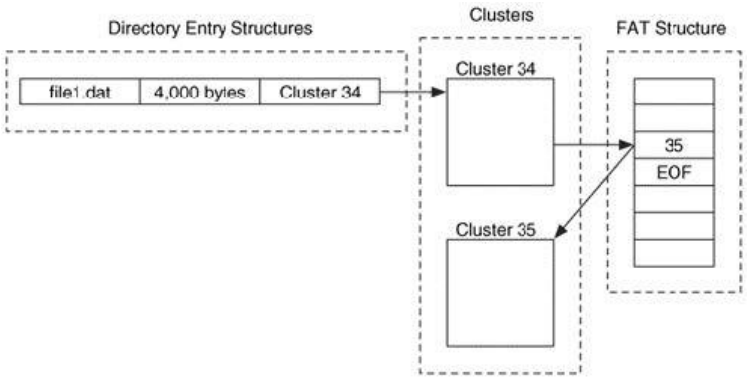
Слика 4.1. – Везе између пет категорија података
(File System Forensic Analysis, Brian Carrier – Figure 8.1.)

Категорија апликација садржи податке који пружају посебне функције. Ови подаци нису потребни током процеса читања или писања фајла и, у многим случајевима, не морају бити укључени у спецификацију фајл система. Ови подаци су укључени у спецификацију јер може бити ефикасније имплементирати их у фајл систем уместо у обичан фајл. Примери података у овој категорији укључују статистику корисничких ограничења и дневнике фајл система. Ови подаци могу бити корисни током истраге, али пошто нису потребни за писање и читање фајла, могу бити лакше фалсификовани него други подаци.

4.1. FAT фајл систем

FAT (File Allocation Table) фајл систем је један од најједноставнијих фајл система присутних у уобичајеним оперативним системима. FAT је примарни фајл систем у Microsoft DOS и Windows 9x оперативним системима, али су NT, 2000, и XP линије прешле на New Technology (NTFS). FAT је подржан од стране свих Windows и већине Unix оперативних система и биће у употреби још годинама, чак и ако није подразумевани фајл систем садашњих desktop Windows система. Предности FAT фајл система укључују једноставност имплементације, подршку за различите оперативне системе (посебно старије верзије Windows-а и Unix-а) и широку компатибилност са различитим уређајима као што су флеш картице и USB дискови.Ипак, FAT фајл систем има неколико недостатака, укључујући ограничење у величини фајлова и партиција, слабу подршку за безбедност и недостатак напредних функционалности које се налазе у модернијим фајл системима као што је NTFS.Управкос својим ограничењима, FAT фајл систем и даље остаје широко присутан у свету технологије, поготово у ситуацијама где је потребна једноставност и компатибилност са различитим платформама.

Један од разлога зашто се FAT фајл систем сматра једноставним је што има мали број типова структура података. На жалост, то такође значи да су током година вршене измене да би му се додавале нове функције. Постоје две битне структуре података у FAT-у (File Allocation Table и директоријумска структура) које служе за више намена и припадају више категоријама модела. FAT фајл систем не садржи никакве податке који припадају категорији апликација.



Слика 4.2. – Однос између ставки директоријумске структуре, кластера и FAT структуре (File System Forensic Analysis, Brian Carrier – Figure 9.1.)

Основна концепција FAT фајл система је да се сваком фајлу и директоријуму додели структура података, ставка директоријумске структуре, која садржи:

- име фајла,
- величину,
- почетну адресу садржаја фајла и
- друге метаподатке.

Садржај фајла и директоријума се чува у јединицама података које се зову кластери. Ако је фајл или директоријум додељен више од једног кластера, остали кластери се налазе коришћењем FAT структуре. FAT структура се користи да идентификује следећи кластер у фајлу, а такође се користи и за идентификацију стања доделе кластера. Због тога се користи и у категоријама садржаја датотека и метаподатака.

Постоје три различите верзије FAT-а:

1. FAT12,
2. FAT16 и
3. FAT32.

Распоред FAT фајл система има три физичка дела, која се могу видети на наредној слици. Први део је резервисано подручје, и укључује податке у категорији фајл система. У FAT12 и FAT16, ово подручје је обично величине само 1 сектор, али тачна величина је дефинисана у boot сектору. Други део је FAT подручје, и садржи основне и резервне FAT структуре. Почиње у сектору који следи након резервисаног подручја, а његова величина се рачуна на основу броја и величине FAT структура. Трећи део је подручје података, и садржи кластере који ће бити додељени за смештање садржаја фајлова и директоријума.

Слика 4.3. – Физичка организација FAT фајл система
(File System Forensic Analysis, Brian Carrier – Figure 9.2.)

Да бисмо дошли до фајла који се састоји од више блокова, као што је „/dir1/dir2/file.txt“, користимо информације из FAT табеле и директоријумске структуре (directory entry). Прво, проверавамо структуру директоријума „/dir1“ како бисмо пронашли ставку за „/dir2“. Сваки директоријум има своју ставку у којем се налазе име поддиректоријума или фајлова и њихова адреса у меморији (кластеру). На пример, ако постоји ставка за „/dir2“ у „/dir1“, пронаћи ћемо информације о томе где се „/dir2“ налази у меморији.

Затим, директоријумска структура за „/dir2“ ће нам дати информације о томе где се фајл „file.txt“ налази. У овој ставци, пронаћи ћемо име фајла (file.txt) и његову адресу у меморији, која показује на почетак листе блокова који чине садржај фајла.

Када имамо почетну адресу фајла (file.txt), можемо се обратити FAT табели. FAT табела је као индекс који нам омогућава да пратимо низ блокова који чине одређени фајл. Првобитно, у ставци директоријумске структуре за file.txt наћи ћемо прву локацију блока који садржи део садржаја фајла. Фајл се састоји од блокова који су улачани тако да чини ланчану листу. Затим, у FAT табели, пронаћи ћемо следећи блок који чини фајл на основу адресе првог блока. Овај поступак понављамо све док не дођемо до последњег блока који садржи крај фајла.

На тај начин, комбиновањем информација из структуре директоријума и FAT табеле, можемо пратити путању до фајла који се састоји од више блокова, као што је „/dir1/dir2/file.txt“, и реконструисати његов садржај из блокова на диску.

Резервисано подручје почиње у сектору 0 фајл система, а његова величина је наведена у boot сектору. За FAT12/16 резервисано подручје је обично само 1 сектор, али FAT32 обично резервише више сектора. FAT подручје садржи једну или више FAT структура, и почиње у сектору након резервисаног подручја. Његова величина се израчунава множењем броја FAT структура са величином сваке FAT; обе ове вредности су наведене у boot сектору. Област података садржи кластере који ће садржати садржај фајлова и директоријума и почиње у сектору након FAT области. Његова величина се израчунава одузимањем почетне адресе сектора подручја података од укупног броја сектора у фајл систему, који је наведен у boot сектору. Подаци су организовани у кластере, а број сектора по кластеру је наведен у boot сектору. Распоред подручја података је мало другачији у FAT12/16 и FAT32. У FAT12/16 почетак подручја података је резервисан за root директоријум, али у FAT32 root директоријум може бити било где у подручју података (иако је ретко да то није на почетку подручја података). Динамична величина и локација root директоријума омогућава FAT32 да се прилагоди лошим секторима на почетку подручја података и омогућава директоријуму да расте колико год је потребно. Root директоријум у FAT12/16 има фиксну величину која је наведена у boot сектору. Почетна адреса за FAT32 root директоријум је наведена у boot сектору, а FAT структура се користи за одређивање његове величине.

Поред информација о распореду, boot сектор садржи многе не толико значајне вредности. Неважне вредности су оне које нису потребне за фајл систем да сачува и поврати фајлове, и оне су ту како би пружиле додатне информације, али можда нису тачне.

Прва три бајта boot сектора садрже инструкцију за скок, у машинском коду, која наводи CPU да прескочи конфигурационе податке и настави са извршавањем остатка boot кода. Boot сектор је величине 512 бајтова, а бајтови од 62 до 509 у FAT12/16 и бајтови од 90 до 509 у FAT32 се не користе. Ови бајтови садрже boot код, а FAT32 може користити секторе који следе после boot сектора за додатни boot код.

Често се дешава да FAT фајл системи имају boot код иако нису покретни фајл системи. Овај boot код обично приказује поруку која упозорава да је потребан други диск за покретање система. Иако може изгледати контрадикторно да фајл систем који није намењен покретању садржи boot код, то је корисно јер омогућава неки облик упозорења корисницима који покушавају да покрену систем са неисправног диска. FAT boot код се позива из boot кода у Master Boot Record-у (MBR) диска. MBR је први сектор на диску и садржи мали програмски код који је одговоран за иницијализацију система при подизању рачунара. Када се рачунар покрене, MBR се прво учита, а затим се покреће boot код унутар MBR-а. Након што је FAT boot код позван из MBR-а, његова функција је да пронађе и учита одговарајуће фајлове оперативног система са диска. То обично укључује тражење OS фајлова који су смештени на партицији која користи FAT фајл систем. Када се пронађу потребни OS фајлови, они се учитавају у меморију и покрећу како би се наставио процес подизања система.

4.1.1 Скривање података у FAT фајл систему

Постоје неколико места која нису коришћена од стране фајл система, и могла би да садрже податке које је корисник сакрио. На пример, постоји преко 450 бајтова података између краја података о boot сектору и ознаке о крају сектора. Windows обично користи овај простор за чување boot кода за систем, али није потребан за фајл системе који нису намењени за подизање система.

FAT32 фајл системи обично алоцирају много сектора за резервисано подручје, али само неколико се користи за примарни boot сектор, резервни boot сектор и FSINFO структуру података. Стога, ова подручја би могла да садрже скривене податке. Осим тога, FAT32 FSINFO структура података има стотине неискоришћених бајтова. Оперативни систем обично брише секторе у резервисаном подручју приликом креирања фајл система.

Такође могу постојати скривени подаци између краја фајл система и краја волумена. Да би се пронашао неискоришћени простор на крају потребно је упоредите број сектора у фајл систему, који је дат у боот сектору, са бројем сектора који су на волумену. Релативно је лако некоме да направи неискоришћени простор на волумену јер им је потребно да измене само укупан број сектора у боот сектору.

FAT32 фајл системи имају резервни boot сектор, и требало би да се налази у сектору 6. Примарна и резервна копија могу се упоредити како би се идентификовале неусаглашености. Ако је примарна копија оштећена, треба прегледати резервну. Ако је корисник изменио неке ознаке или друге вредности у примарном boot сектору коришћењем hex едитора, резервна копија може садржати оригиналне податке.

4.2. NTFS фајл систем

New Technology File System (NTFS) је дизајниран од стране Microsoft-а и подразумевани је фајл систем за оперативне системе Microsoft Windows NT, Windows 2000, Windows XP, and Windows Server. FAT ће и даље постојати у мобилним и малим уређајима за смештање података, али је NTFS вероватно најчешћији фајл систем за Windows системе. NTFS је значајно комплекснији фајл систем него FAT због тога што има многе функције и веома је расположив.

NTFS је дизајниран за поузданост, безбедност и подршку за велике уређаје за смештање података. Расположивост је обезбеђена употребом генеричких структура података које обухватају структуре података са конкретним садржајем. Ово је дизајн који има могућност скалирања јер се интерна структура података може мењати током времена када се уводе нови захтеви фајл систему, а општи wrapper може остати константан. Један пример генеричког wrapper-а је да је сваки бајт података у NTFS фајл систему алоциран за фајл. Ово значи да сваки бајт података у фајл систему, било да је реч о подацима из датотека или метаподацима, мора бити придружен одговарајућем фајлу. То омогућава велику флексибилност и кључно управљање

подацима у фајл систему, као и бољу организацију и контролу над подацима. На пример, овај приступ олакшава брз приступ подацима, јер се сваки податак може лако лоцирати и идентификовати путем његовог придруженог фајла.

NTFS је комплексан фајл систем и, на жалост, нема објављене спецификације од стране Microsoft-а која описује распоред података на диску. Описи на високом нивоу компоненти фајл система су објављени, али детаљи описи су ретки.

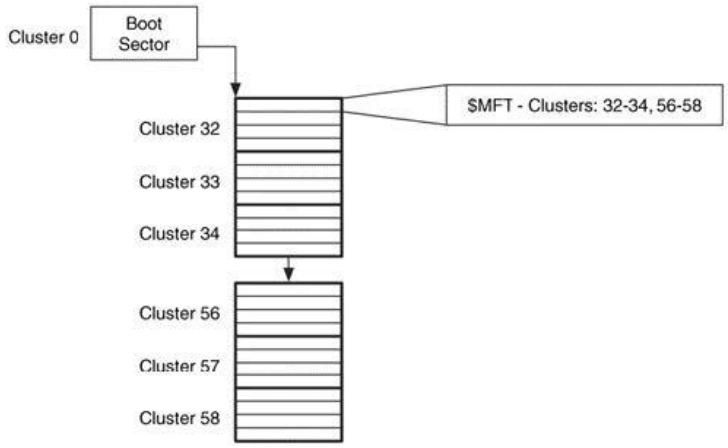
Један од најважнијих концепата у разумевању дизајна NTFS-а је да се важни подаци алоцирају као фајлови. Ово укључује основне административне податке фајл система који су обично сакривени код других фајл система. Заправо, фајлови који садрже административне податке могу бити смештени било где на волумену, као што је то случај са обичним фајловима. Стога, NTFS фајл систем нема специфичан распоред као други фајл системи. Цео фајл систем се сматра облашћу података, и сваки сектор може бити додељен фајлу. Једини конзистентни распоред је да први сектори волумена садрже boot сектор и boot код.

4.2.1. Master File Table структура

Master File Table (MFT) је главна компонента NTFS-а јер садржи информације о свим фајловима и директоријумима. Сваки фајл и директоријум имају барем једну ставку у табели, а саме ставке су веома једноставне. Величине су 1 KB, али само прва 42 бајта имају дефинисану сврху. Преостали бајтови чувају атрибуте, који су мале структуре података са веома специфичном сврхом. На пример, један атрибут се користи за чување имена фајла, док се други користи за чување садржаја фајла.

Microsoft назива сваку ставку у табели *file record*. Свака ставка добија адресу на основу своје локације у табели, почевши од 0. Сви уноси су величине 1.024 бајта, али тачна величина је дефинисана у boot сектору.

Као и све у NTFS-у, MFT је фајл. Оно што ово чини збуњујућим је то што MFT има унос за себе. Први унос у табели се назива $MFT, и описује локацију MFT-а на диску. Заправо, то је једино место где је описана локација MFT-а; стога је потребно обрадити га да бисте одредили распоред и величину MFT-а. Почетна локација MFT-а је дата у boot сектору, који се увек налази у првом сектору фајл система. Можемо видети то на наредној слици где се boot сектор користи за проналажење првог уноса MFT-а, што показује да је MFT фрагментиран и простире се од кластера 32 до 34 и од 56 до 58. Као и FAT, NTFS користи кластере, који су групе узастопних сектора.



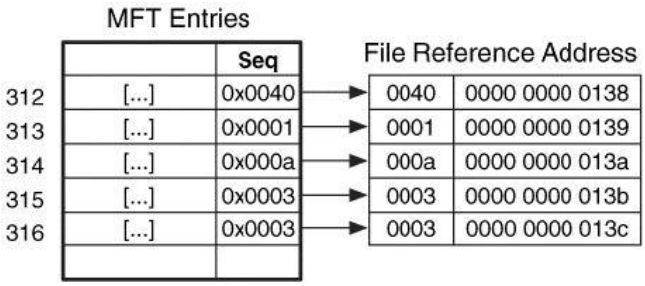
Слика 4.4. – Релација између boot сектора и $MFT структуре
(File System Forensic Analysis, Brian Carrier – Figure 11.2.)

У Microsoft-овој имплементацији NTFS-а, MFT почиње што је могуће мањи и проширује се када су потребни додатни уноси. Теоријски, оперативни систем би могао да креира фиксни број уноса приликом креирања фајл система, али динамична природа Microsoft-ове имплементације омогућава му лако проширивање фајл система када се додаје више простора од преклапања волумена. Microsoft не брише уносе MFT-а након што су креирани.

Величина сваког MFT ставке је дефинисана у boot сектору, али све верзије од Microsoft-а користиле су величину од 1.024 бајта. Првих 42 бајта структуре података садржи 12 поља, а преосталих 982 бајта су неструктурирани и могу бити попуњени атрибутима.

Прво поље у сваком MFT запису је потпис, које код стандардних записа има вредност ASCII низ карактера "FILE." Ако се пронађе грешка у запосу, може имати вредност "BAAD." Постоји и поље за flag-ове које идентификује да ли се ставка табеле користи и да ли се та ставка односи на директоријум. Статус алокације MFT ставке такође се може одредити из $BITMAP атрибута у $MFT фајлу. Ако атрибути фајла не могу да стану у једану ставку табеле, могу се користити више ставки. Када се то деси, прва ставка се зове *base file record*, или *base MFT enrty* (основна MFT ставка), а свака од наредних ставки садржи адресу основне ставке у једном од својих фиксних поља.

Свака MFT ставка је секвенцијално адресирана коришћењем 48-битне вредности, и прва ставка има адресу 0. Максимална MFT адреса се мења како MFT расте и одређује се дељењем величине $MFT са величином сваког уноса. Microsoft овај секвенцијални адресар назива бројем фајла. Свака MFT ставка такође има 16-битни секвенцијални број који се повећава када се ставка алоцира. На примеру са наредне слике, ставка 313 из MFT табеле има секвенцијални број 1. Фајл који је алоцирао ставку 313 је избрисан, и та ставка је поново алоцирана новом фајлу. Када је ставка поново алоцирана, она има нови секвенцијални број 2. MFT ставка и секвенцијални број се комбинују, са секвенцијалним бројем у горњих 16-бита, тако да формирају 64-битну адресу – *file reference address.*



Слика 4.5. – Формирање file reference адресе комбинацијом адресе ставке и секвенцијалног броја (File System Forensic Analysis, Brian Carrier – Figure 11.3.)

NTFS користи адресу референце фајла да референцира MFT ставке јер секвенцијални број олакшава одређивање када је фајл систем у корумпираном стању.

1. Идентификација валидности: Када се унос у MFT поново алоцира новом фајлу, његов секвенцијални број се повећава. Ако нека структура података садржи MFT адресу, можемо проверити секвенцијални број да видимо да ли је та структура и даље важећа за тај фајл. Ако се број не поклапа, значи да је унос поново алоциран другом фајлу и структура података је застарела.
2. Детекција корупције: Ако систем падне док се фајл креира или модификује, можда ће остати делимично попуњене или неконзистентне структуре података. Секвенцијални број може да помогне да се утврди да ли су ове структуре података у складу са тренутним стањем MFT уноса. Ако секвенцијални бројеви не одговарају, то је знак да је дошло до корупције и да су структуре података можда непоуздане.
3. Опоравак избрисаних фајлова: Када се фајл избрише, његов MFT унос може бити поново алоциран новом фајлу. Ако пронађемо неалоцирану структуру података која садржи референцу на MFT унос, можемо користити секвенцијални број да проверимо да ли је тај MFT унос у међувремену поново алоциран. Ако је секвенцијални број промењен, знамо да је MFT унос сада у употреби од стране новог фајла и стара структура података више није важећа.

4.2.2. NTFS метаподаци

Пошто је сваки бајт у волумену алоциран неком фајлу, морају постојати фајлови који чувају административне податке фајл система. Microsoft ове фајлове назива метаподаци фајловима *(metadata files).*

Првих 16 уноса у Master File Table (MFT) су резервисани за метаподаци фајлове. Ови уноси садрже кључне информације које фајл систем користи за управљање својим структурамама и подацима. На пример, ту спадају фајлови као што су $MFT (који описује сам MFT), $LogFile (који садржи податке о дневницима активности) и други.

Уноси који су резервисани али нису активни су у алоцираном стању са основним информацијама. Ово осигурава да су ти уноси припремљени за будућу употребу ако буде потребно.

Фајлови метаподатака су скривени од просечног корисника да би се спречиле случајне измене или брисање критичних системских података. Корисници са одговарајућим привилегијама могу приступити овим фајловима ради одржавања или дијагностике.

Имена метаподаци фајлова почињу са знаком "$" и имају велико прво слово. Ово је конвенција која олакшава њихово препознавање и разликовање од других фајлова.

Стандардни фајлови метападатака у NTFS фајл систему:

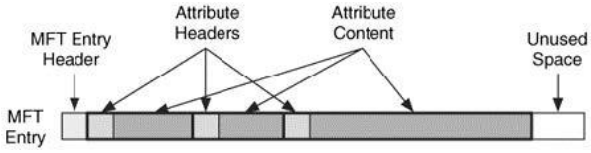
1. $MFT – фајл за саму MFT табелу. $MFT је најважнији фајл у NTFS фајл систему. Садржи записе о свим фајловима и директоријумима на волумену. Сваки фајл и директоријум има најмање један унос у $MFT-у.
2. $MFTMirr – садржи копију првих 4 уноса из $MFT-а ради редунданције и опоравка у случају корупције главног $MFT фајла. Овај фајл помаже у одржавању интегритета фајл система.
3. $LogFile – чува записе о свим трансакцијама које се дешавају у фајл систему. Користи се за опоравак у случају пада система или неке друге грешке. Садржи информације о операцијама које су у току и осигурава да се све трансакције правилно заврше или пониште.
4. $Volume – садржи опште информације о волумену, као што су име волумена, верзија фајл система, и време када је волумен створен. Овај фајл не садржи податке о фајловима или директоријумима, већ опште метаподатке.
5. $AttrDef – садржи дефиниције свих атрибута који могу бити додељени фајловима и директоријумима у NTFS-у. Овај фајл дефинише типове атрибута, њихове ознаке и друга својства.
6. . – представља root директоријум волумена. Садржи уносе за све директоријуме и фајлове који се налазе у корену фајл система. Овај фајл је почетна тачка за све путање у фајл систему.
7. $Bitmap – прати који су кластери у волумену алоцирани, а који су слободни. Битмапа користи битове за означавање статуса кластера, где један бит представља један кластер.
8. $Boot – садржи boot сектор који се користи за подизање оперативног система. Овај сектор садржи код и податке неопходне за иницијализацију система приликом покретања.
9. $BadClus – бележи све лоше (оштећене) кластере у волумену. Овај фајл осигурава да се оштећени кластери не користе за чување података.
10. $Secure – фајл чува информације о безбедносним дескрипторима који одређују дозволе приступа за фајлове и директоријуме. Ово укључује податке о власништву, дозволама и другим безбедносним подешавањима.
11. $Upcase – фајл садржи табелу која се користи за конверзију малих слова у велика слова. Ова табела је корисна за операције које су неосетљиве на разлику између великих и малих слова.
12. $Extend – садржи додатне фајлове метаподатака који проширују функционалност NTFS-а. Ови фајлови могу укључивати додатне атрибуте, информације о ограничењима и друге проширења која нису директно подржана основним фајл системом.

4.2.3. NTFS атрибути

Ставка MFT табеле има мало унутрашње структуре и већином је коришћен за чување атрибута, који су структуре података које чувају одређене информације. Постоји много типова атрибута, и сваки има своју унутрашњу структуру. На пример, постоје атрибути за име фајла, датум и време, па чак и за његов садржај. Ово је један од начина на који се NTFS веома разликује од других фајл система. Већина фајл система чита и уписује сам садржај фајла, али NTFS чита и уписује атрибуте, од којих један садржи садржај фајла.

Иако сваки тип атрибута чува различит тип података, сви атрибути имају два дела: заглавље и садржај. Заглавље је генеричко и стандардно за све атрибуте. Садржај је специфичан за тип атрибута и може бити било које величине.

Заглавље атрибута идентификује тип атрибута, његову величину и његово име. Такође има flag-ове које идентификују да ли је вредност компресована или енкриптована. Тип атрибута је нумерички идентификатор заснован на типу података. Унутар једне MFT ставке може бити више атрибута истог типа.

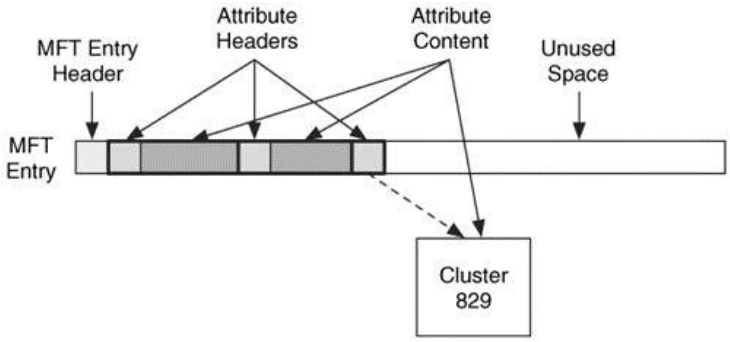


Слика 4.6. - MFT ставка са назначеним заглављем и садржајем
(File System Forensic Analysis, Brian Carrier – Figure 11.4.)

Неки атрибути могу имати додељено име које се чува у UTF-16 Unicode формату у заглављу атрибута. Атрибут такође има идентификациони број који му је додељен и који је јединствен за ту MFT ставку. Ако ставка има више атрибута истог типа, овај идентификатор може бити коришћен за разликовање између њих.

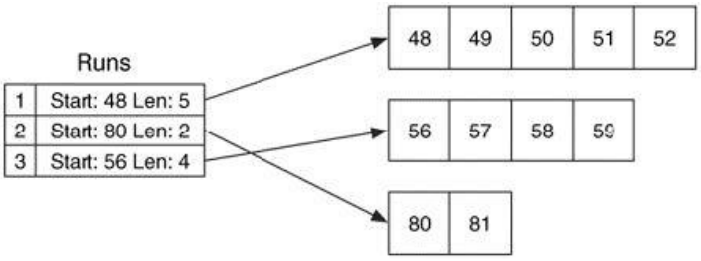
Садржај атрибута може имати било који формат и било коју величину. На пример, један од атрибута се користи за чување садржаја фајла, што може бити неколико мегабајта или гигабајта. Није практично чувати ову количину података у MFT запису који има само 1.024 бајта.

Да би се решио овај проблем, NTFS обезбеђује два места где се садржај атрибута може чувати. Резидентни атрибут (*resident attribute*) чува свој садржај у MFT запису заједно са заглављем атрибута. Ово функционише само за мале атрибуте. Нерезидентни атрибут (*non-resident attribute*) чува свој садржај у спољном кластеру у фајл систему. Заглавље атрибута одређике да ли је атрибут резидентан или нерезидентан. Ако је атрибут резидентан, садржај ће одмах следити заглавље. Ако је атрибут нерезидентан, заглавље ће дати адресе кластера где се садржај налази.



Слика 4.7. – Пример резидентних и нерезидентних атрибута
(File System Forensic Analysis, Brian Carrier – Figure 11.5.)

Нерезидентни атрибути се чувају у низовима кластера, који су узастопни кластери, и тај низ је одређен на основу почетне адресе кластера и дужине низа. Ако су атрибуту додељени кластери 48, 49, 50, 51 и 52, он има низ који почиње у кластеру 48 и дужине је 5 кластера. Ако су атрибуту такође додељени кластери 80 и 81, он има други низ који почиње у кластеру 80 и дужине је 2 кластера. Трећи низ може почети у кластеру 56 и имати дужину од 4 кластера.



Слика 4.8. – Пример низа кластера у коме се смешта садржај атрибута
(File System Forensic Analysis, Brian Carrier – Figure 11.6.)

Логичка адреса фајл система је дефинисана као адреса додељена јединицама података фајл система, а логичку адресу фајла као адресу релативну у односу на почетак фајла. NTFS користи различите термине за ове адресе. Логички број кластера (*LCN, Logical Cluster Number*) је исти као логичка адреса фајл система, а виртуелни број кластера (*VCN, Virutal Cluster Number*) је исти као логичка адреса фајла. NTFS користи мапирање VCN-а у LCN да опише низове нерезидентних атрибута. Ако се вратимо на претходни пример, низ овог атрибута показује да се VCN адресе од 0 до 4 мапирају на LCN адресе од 48 до 52, VCN адресе од 5 до 6 се мапирају на LCN адресе од 80 до 81, и VCN адресе од 7 до 10 се мапирају на LCN адресе од 56 до 59.

За сваки тип атрибута дефинисан је број који га одређује. Microsoft сортира атрибуте у једној ставки користећи овај број. Стандардни атрибути имају подразумевану вредност типа која им је додељена, али се она касније може редефинисати у $AttrDef фајлу метаподатака. Поред броја, сваки тип атрибута има и име, које се пише великим словима и почиње симболом "$".

Идентификатор типа	Име	Опис

16	$STANDARD_INFORMATION	Садржи опште информације као што су flag-ови; последње време приступа, писања и креирања; и идентификатор власника и безбедности
32	$ATTRIBUTE_LIST	Листа где се могу наћи други атрибути за фајл
48	$FILE_NAME	Име фајла, у Unicode формату, и последње време приступа, писања и креирања
64	$VOLUME_VERSION	Информације о верзији волумена. Постоји само у верзији 1.2 (Windows NT)
64	$OBJECT_ID	Јединствени 16-бајтни идентификатор за фајл или директоријум. Постоји само у верзијама 3.0+ и касније (Windows 2000+)
80	$SECURITY_DESCRIPTOR	Особине контроле приступа и сигурности фајла
96	$VOLUME_NAME	Име волумена
112	$VOLUME_INFORMATION	Верзија фајл система и други flag-ови
128	$DATA	Садржај фајла
144	$INDEX_ROOT	Коренски чвор индексног стабла
160	$INDEX_ALLOCATION	Чворови индексног стабла у $INDEX_ROOT атрибуту
176	$BITMAP	Битмапа за $MFT фајл и за индексе
192	$SYMBOLIC_LINK	Информације о soft линковима. Постоји само у верзији 1.2 (Windows NT)
192	$REPARSE_POINT	Садржи податке о reparse point-у, који се користи као soft линк у верзијама 3.0+ (Windows 2000+)
208	$EA_INFORMATION	Користи се за backward компатибилност са OS/2 апликацијама (HPFS)
224	$EA	Користи се за backward компатибилност са OS/2 апликацијама (HPFS)
256	$LOGGED_UTILITY_STREAM	Садржи кључеве и информације о шифрованим атрибутима у верзијама 3.0+ (Windows 2000+)

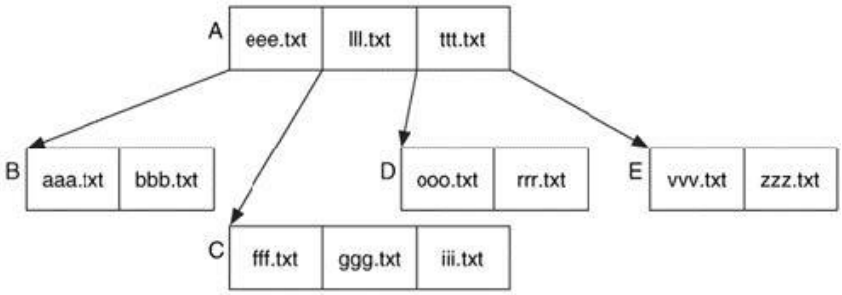
Табела 4.1. – Листа подразумеваних атрибута у MFT ставкама

4.2.4. Интекси

NTFS користи индексне структуре података у многим ситуацијама. Индекс у NTFS-у је колекција атрибута који су поређани у сортираном редоследу. Најчешћа употреба индекса је у директоријумима, јер директоријуми садрже $FILE_NAME атрибуте.

NTFS користи B-стабла, која су слична бинарним стаблима, али могу имати више од два детета по чвору. Обично, број деце које чвор има зависи од тога колико вредности сваки чвор може да складишти.

Чвор A садржи три вредности и четири детета. Ако тражимо фајл "ggg.txt", погледали бисмо вредности у коренском чвору и утврдили да име алфабетски спада између "eee.txt" и "lll.txt". Према томе, прелазимо на чвор C и погледамо његове вредности. На тај начин проналазимо име фајла у том чвору.



Слика 4.9. – Пример B-tree структуре
(File System Forensic Analysis, Brian Carrier – Figure 11.6.)

4.3. Ext2 и Ext3 фајл системи

Фајл системи Ext2 и Ext3, који се често заједно називају ExtX, су подразумевани фајл системи за многе дистрибуције Linux оперативног система. Ext3 је новија верзија Ext2 и додаје journaling, али основна конструкција Ext2 остаје иста. ExtX су базирани на UNIX фајл систему (UFS). ExtX су уклонили многе компоненте UFS-а које више нису потребне, тако да су једноставнији за разумевање и објашњење. За разлику од других оперативних система, Linux подржава велики број фајл система, и свака дистрибуција може одабрати који ће бити подразумевани.

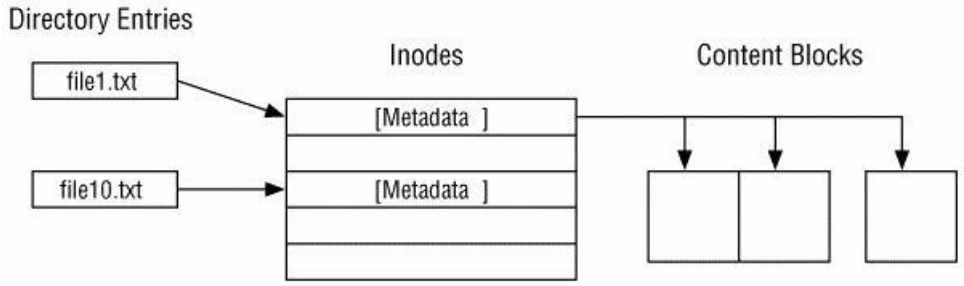
ExtX узима свој дизајн из UFS-а, који је дизајниран да буде брз и поуздан. Копије важних структура података су похрањене широм фајл система, а сви подаци повезани са фајлом су локализовани како би се смањила потреба за померањем глава хард диска приликом њиховог читања.

Фајл систем почиње са опционом резервисаном облашћу, а остали део фајл система је подељен на блок групе. Све блок групе, осим последње, садрже исти број блокова, који се користе за складиштење имена фајлова, метаподатака и садржаја фајлова.

Основне информације о распореду ExtX су смештене у суперблок структури података, која се налази на почетку фајл система. Друге значајне структуре су блокови, inode и

директоријумска структура (*directory entry structure*). Садржај фајлова је смештен у блоковима, који представљају групе узастопних сектора. Метаподаци за сваки фајл и директоријум су смештени у структури података која се зове инод (*inode*). Inode-и су фиксне величине и налазе се у таблици inode-ова. Постоји једна табела inode-ова у свакој блок групи. Име фајла је смештено у директоријумској структури, која се налази у блоковима додељеним родитељском директоријуму фајла. Ове структуре су једноставне и садрже име фајла и показивач на inode фајла.

Разлика између Ext2 и Ext3 система је то што Ext2 не подржава journaling и користи се у ситуацијама када је брзина битнија од отпорности на грешке. Ext3 додаје journaling, повећава отпорност на грешке и брже опорављање након пата система.



Слика 4.10. – Односи између ставки директоријума, inode структуре и блокова
(File System Forensic Analysis, Brian Carrier – Figure 14.1.)

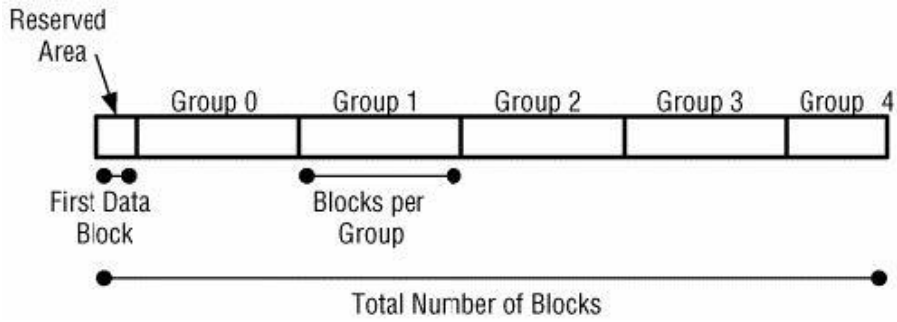
ExtX има опционе функције организоване у три категорије на основу тога како оперативни систем треба да се понаша ако наиђе на фајл систем са функцијом коју не подржава. Прва категорија су компатибилне функционалности, и ако оперативни систем не подржава неку од ових функција, она ће ипак моћи да подеси фајл систем и настави са радом као и обично. Примери укључују методе за алокацију, постојање журнала фајл система и проширених атрибута. Постоје и не компатибилне функционалности, и ако оперативни систем наиђе на неку од њих, не би требало да прихвата такав фајл систем. Пример овога укључује компресију. Коначно, функционалност може бити компатибилна за читање само. Када оперативни систем наиђе на једну од ових функционалности коју не подржава, фајл систем треба бити подешен само за читање. Примери овога укључују подршку за велике фајлове и коришћење B-стабала за сортирање директоријума уместо несортиране листе.

У ExtX-у, постоје две структуре података које чувају податке у категорији фајл система: суперблок и дескриптор групе. Суперблок се налази на почетку фајл система и садржи основне информације о величини и конфигурацији. Сличан је структурама података заглавља у структурама NTFS или FAT фајл система. Као што је раније поменуто, фајл систем је организован у блок групе, и свака група има дескриптор групе који описује распоред групе. Деск риптори група се налазе у табели дескриптора групе, која се налази у блоку после суперблока. Резервне копије суперблока и табеле дескриптора групе постоје широм фајл система у случају оштећења примарних копија.

4.3.1. Суперблок и дескриптори блок групе

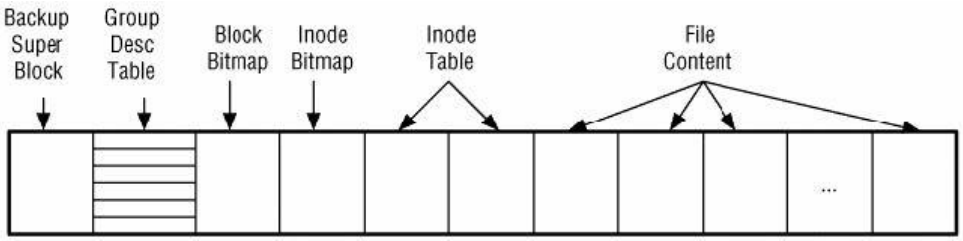
Суперблок у ExtX фајл систему се налази на 1,024 бајта од почетка фајл система и има величину од 1,024 бајта, иако већина бајтова није употребљена. Ова структура података садржи само конфигурационе вредности и не садржи заглавље. Резервне копије суперблока обично се чувају у првом блоку сваке блок групе.

Суперблок садржи основне информације, као што су величина блока, укупан број блокова, број блокова по блок групи и број резервисаних блокова пре прве блок групе. Такође садржи укупан број inode структура и број inode-а по блок групи. Необавезни подаци у суперблоку укључују име волумена, последње време писања, последње време монтирања и путању где је фајл систем последњи пут монтиран.



Слика 4.11. – Распоред 5 блок група у ExtX фајл систему
(File System Forensic Analysis, Brian Carrier – Figure 14.2.)

Суперблок такође чува неке податке о укупном броју слободних inode-а и блокова. Ови подаци се користе када се алоцирају нови inode-и и блокови. Да би се одредио распоред фајл система, прво користимо величину блока и број блокова да бисмо израчунали величину фајл система. Ако је ова вредност мања од величине волумена, могу постојати скривени подаци који следе након фајл система. Прва блок група се налази у блоку који следи након резервисаног подручја.



Слика 4.12. – Изглед блок групе
(File System Forensic Analysis, Brian Carrier – Figure 14.3.)

У блоку који следи након суперблока је табела дескриптора групе, која садржи дескриптор структуре за сваку блок групу у фајл систему. Резервне копије табеле постоје у свакој од блок група, осим ако је омогућена функционалност sparse суперблока. Поред садржаја фајлова, блок групе садрже административне податке, као што су суперблокови, табеле дескриптора групе, табеле inode-а, битмапе inode-а и битмапе блокова. Дескриптор групе описује где се ови подаци могу наћи.

Битмапа блокова управља статусом алокације блокова у групи, а његова почетна адреса блока је наведена у дескриптору групе. Величина битмапе у бајтовима може се

израчунати дељењем броја блокова у групи са осам. Када Linux креира фајл систем, дефинише број блокова по групи тако да буде једнак броју битова у блоку. Стога ће битмапа блокова захтевати тачно један блок.

Битмапа inode-а управља статусом алокације inode-а у групи, а његова почетна адреса блока такође је наведена у дескриптору групе. Величина битмапе у бајтовима може се израчунати дељењем броја inode-а по групи са осам. Уопштено, постоји мање inode-а него блокова по групи, али корисник може изабрати ове вредности када креира фајл систем. Најзад, почетна адреса блока inode-а табеле наведена је у дескриптору групе, а њена величина се рачуна множењем броја inode-а по групи са величином сваког inode-а, која износи 128 бајтова.

Дескриптор групе такође садржи број слободних блокова и inode-а у блок групи. Суперблок садржи укупан број слободних блокова и inode-а у свим групама.

Ако ExtX фајл систем садржи оперативни систем, онда он садржи и секцију за boot код. Сви остали фајл системи који нису boot-абилни не треба да садрже boot code секцију. Када она постоји, то је првих 1,024 бајта испред суперблока (то јест, прва два сектора). Boot code ће бити извршен након што контрола пређе на њега из boot code-а у Master Boot Record-у (MBR) сектору 0 диска. ExtX boot code обично зна који су блокови алоцирани језгру и учитаће их у меморију.

4.3.2. Inode структуре

У ExtX, основни метаподаци фајла се чувају у inode структури података. Додатни метаподаци могу бити смештени у проширене атрибуте. Све inode структуре у ExtX-у имају исту величину, која се може дефинисати у суперблоку. Један inode се алоцира за сваки фајл и директоријум, а сваки inode има адресу, која почиње од 1. За сваку групу блокова додељен је сет inode-ова, чија величина је наведена у суперблоку. Inode-и у свакој групи се чувају у табели, чија је локација наведена у дескриптору групе. Уколико је дата адреса inode-а, његова група може се одредити следећим израчунавањем:

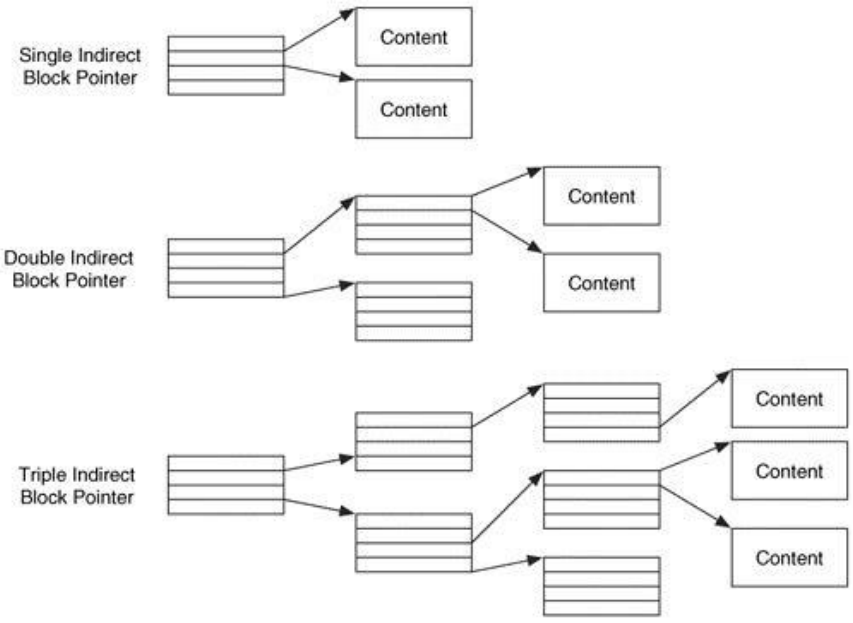
$$group = (inode - 1) / inode\_per\_group.$$

Inode-и од 1 до 10 су обично резервисани и требало би да буду у алоцираном стању. Суперблок има вредност првог нерезервисаног inode-а. Од резервисаних inode-а, само inode 2 има специфичну функцију и користи се за root директоријум. Inode 1 прати лоше блокове, али нема никакав специјалан статус у језгру Линукса. Журнал обично користи inode 8, али ово се може префинити у суперблоку. Први кориснички фајл обично се алоцира у inode-у 11, а ово се често користи за *lost+found* директоријум. Checker-и конзистенције фајл система користе *lost+found* директоријум, и сваки inode који је алоциран али нема име фајла које на њега указује, додат је у овај директоријум са новим именом.

Сваки inode има статички број поља, а додатне информације могу бити смештене у продужене атрибуте и индиректне block pointer-е. Статус алокације inode-а одређује се коришћењем битмапе inode-а, чија је локација наведена у дескриптору групе.

ExtX, као и UFS, развијен је са циљем оптимизације за мале фајлове. Због тога, сваки inode може чувати адресе првих 12 блокова које је фајл алоцирао. Ови се зову директни показивачи. Ако фајл захтева више од 12 блокова, алоцира се блок да чува преостале адресе. Показивач на блок се зове индиректни блок показивач. Адресе у блоку су све четири бајта, а укупан број у сваком блоку заснован је на величини блока. Индиректни блок показивач је смештен у inode-у.

Ако фајл има више блокова него што може да стане у 12 директних показивача и индиректном блоку, користи се двоструки индиректни блок. Двоструки индиректни блок представља структуру када inode показује на блок који садржи списак појединачних индиректних блок показивача, од којих сваки показује на блокове који садрже списак директних показивача. Ако фајл и даље захтева више простора, може користити троструки индиректни блок показивач. Троструки индиректни блок садржи адресе двоструког индиректног блока, који садрже адресе једноструких индиректних блокова.



Слика 4.13. – Примери једноструких, двоструких и троструких блок показивача
(File System Forensic Analysis, Brian Carrier – Figure 14.5.)

4.4. UFS1 и UFS2 фајл системи

UFS (UNIX File System), такође познат као FFS (Fast File System), је фајл систем који се користи у многим UNIX и UNIX-попутним оперативним системима. UFS има две главне верзије: UFS1 и UFS2. Ова два система деле многе основне концепте, али се

разликују у неколико кључних аспеката који утичу на перформансе, функционалност и скалабилност. UFS1 фајл систем је подразумевани фајл систем у OpenBSD и Solaris оперативним системима. Био је подразумевани фајл систем у FreeBSD и NetBSD све док FreeBSD 5.0 и NetBSD 2.0 нису укључили UFS2.

UFS1 је прва верзија овог фајл система и уведена је у BSD UNIX почетком 1980-их. Она је значајно побољшала перформансе у односу на старије UNIX фајл системе захваљујући новим техникама расподеле података и метаподатака. UFS2 је уведена у FreeBSD 5.0 и доноси бројна побољшања и нове могућности у односу на UFS1, укључујући подршку за веће фајлове и бољу организацију метаподатака.

Док UFS1 и UFS2 деле многе основне концепте, постоје значајне разлике између њих:

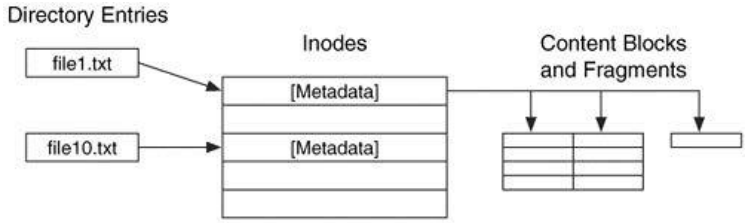
- Величина iNode-а: UFS2 користи веће iNode структуре које укључују додатне временске жигове и друга поља за напредне функције.
- Подршка за веће фајлове: UFS2 подржава веће фајлове и веће укупне величине фајл система у односу на UFS1.
- Боља организација метаподатака: UFS2 пружа побољшану организацију и управљање метаподацима, што резултира бољим перформансама и скалабилношћу.

UFS1 и UFS2 су важни фајл системи у историји UNIX и UNIX-попутних оперативних система, и њихово разумевање пружа основ за проучавање савремених фајл система.

Копије важних структура података смештене су широм фајл система, а подаци су локализовани тако да се магнетне главе храд-дискова не морају много померати приликом читања фајла. UFS је организован у секције, које се називају групе цилиндара (*cylinder groups)*, и величина сваке групе заснива се на геометрији хард-диска. Ове групе су сличне блок групама у ExtX.

4.4.1. UFS суперблокови

UFS има суперблок структуру података на почетку фајл система која садржи основне информације о распореду. Садржај сваког фајла се чува у блоку, који је група узастопних сектора. Блокови такође могу бити подељени на фрагменте, који се користе за чување завршних бајтова фајла уместо да се алоцира цео блок. Метаподаци за сваки фајл и директоријум се чувају у структури података званој inode. Имена фајлова су смештена у директоријумским структурама, које се налазе у блоковима алоцираним за директоријуме. Директоријумске структуре (*directory entry structure*) су основне структуре података које садрже име фајла и показивач на inode структуру фајла. Однос између ових структура података може се видети на следећој слици.\



Слика 4.14. – Однос између директоријумских структура, inode структура и блокова података (File System Forensic Analysis, Brian Carrier – Figure 16.1.)

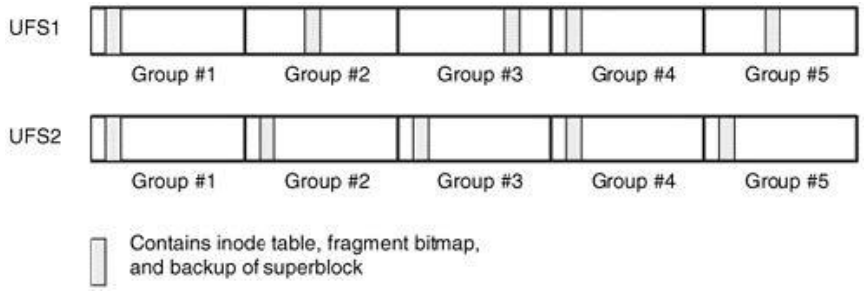
UFS суперблок садржи основне информације, као што су величина сваког фрагмента и број фрагмената у сваком блоку. Овде такође можемо сазнати колико је велика свака цилиндарска група и где се налазе различите структуре података унутар сваке групе. Са овим вредностима можемо утврдити распоред фајл система. Суперблок такође може садржати ознаку волумена и време када је фајл систем последњи пут монтиран. UFS суперблок има исту улогу као суперблок у ExtX, али распоред и неесенцијални подаци су различити.

UFS суперблок се налази негде на почетку фајл система. Код преносивих медијума, може почети у првом сектору. UFS1 суперблок је обично лоциран 8 KB од почетка фајл система, док је UFS2 суперблок обично лоциран 64 KB од почетка. Такође је могуће да UFS2 суперблок постоји на 256 KB од почетка фајл система, али то није подразумевано. Резервне копије суперблока могу се наћи у свакој од цилиндарских група.

4.4.2. UFS цилиндичне групе и дескриптори група

Фајл систем је организован у цилиндарске групе. Све групе, осим можда последње, су исте величине и свака садржи податке дескриптора групе која описује групу. Прва група почиње на почетку фајл система, а број фрагмената у свакој групи дат је у суперблоку. Поред дескриптора групе, свака група такође садржи табелу indoe-а и резервну копију суперблока.

UFS дескриптор групе је много већи од његовог еквивалента у ExtX-у, иако је већи део података неесенцијалан. Дескриптору групе је додељен цео блок и садржи комбинацију стандардних поља и слободног простора који се може користити за различите табеле. Стандардна поља пружају информације о евиденцији и описују како је организован наредни део блока. Информације о евиденцији постоје да би алокација фајлова била ефикаснија, као што је локација последњег додељеног блока, фрагмента и inode-а. Постоје и информације о броју слободних блокова, фрагмената и inode-а, али оне би требало да буду исте као вредности у резимеу цилиндарске групе. Дескриптор групе такође садржи време последњег уписа у групу.

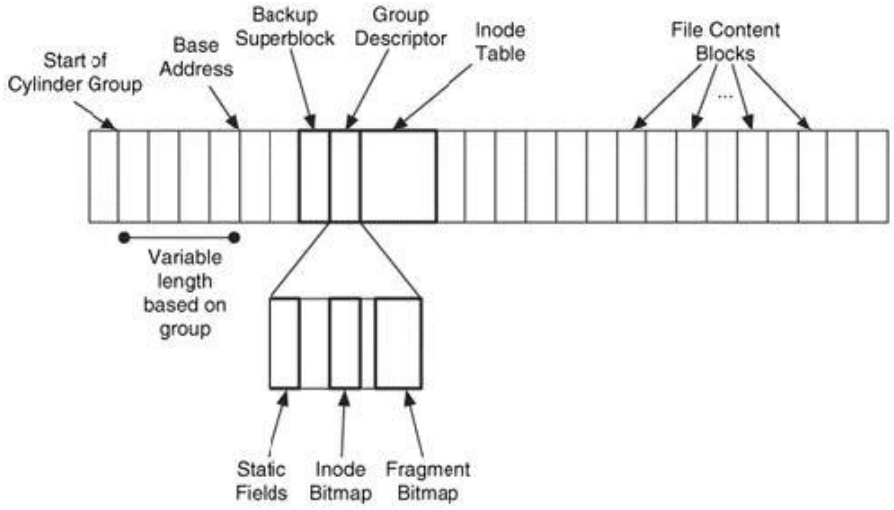


Слика 4.15. – USF1 и UFS2 са пет група и административним подацима
(File System Forensic Analysis, Brian Carrier – Figure 16.3.)

Последњи део дескриптора групе садржи битмапе за indoe-е, блокове и фрагменте у групи. Такође садржи табеле које помажу у проналажењу узастопних фрагмената и блокова одређене величине. Почетна локација сваке од ових структура података дата је као бајт офсет који је релативан у односу на почетак дескриптора групе, а величина структуре података обично мора да се израчуна.

Цилиндричне групе су тако назване зато што су биле поравнате на границе цилиндра. Старији хард-дискови су имали исти број сектора по траци, што је значило да је први сектор сваке групе био на истом слоју. Да би се смањио утицај грешке хард-диска, административни подаци су размакнути тако да свака копија суперблока није на истом слоју. Новији хард-дискови немају исти број сектора у сваком цилиндру, тако да то више није проблем, а UFS2 више не размешта податке.

Базна локација за сваку групу се израчунава коришћењем две вредности из суперблока. Суперблок дефинише циклусну вредност c и делта вредност d. База се повећава за d за сваку групу и враћа се на почетак након c група. На пример, база се може повећавати за 32 фрагмента за сваку групу и онда почети са офсетом од 0 након 16 група.



Слика 4.16. – Изглед USF1 цилиндричне групе
(File System Forensic Analysis, Brian Carrier – Figure 16.3.)

5. Алат за анализу и визуализацију статистике датотека фајл система

Као део овог семинарског рада имплементиран је алат који омогућава анализу и визуелизацију статистике коришћења датотека на фајл систему за одређени временски период. Главни циљ пројекта је омогућити корисницима да боље разумеју како се датотеке користе и приступају на фајл систему, што може бити од велике користи у форензичким истраживањима, истраживачким радовима или оптимизацији система.

Задатак пројекта може се поделити на неколико главних сегмената:

1. Анализа статистике коришћења датотека:
    - Прикупљање информација о приступима датотекама, укључујући време приступа, величину датотеке, тип датотеке (екстензију) и остале релевантне метаподатке.
    - Агрегација прикупљених података како би се генерисала статистика коришћења датотека за специфирани временски период.
2. Визуелизација статистике:
    - Креирање графикона или дијаграма који приказују статистику коришћења датотека по величини, типу и времену приступа.
3. Календарски приказ догађаја у фајл систему:
    - Омогућавање корисницима да одаберу дан и прикажу све догађаје који су се десили у фајл систему у току тог дана. Могуће је праћење четири типа догађаја:
        1. креирање датотеке,
        2. измена датотеке,
        3. преименовање датотеке,
        4. брисање датотеке
4. Кориснички интерфејс:
    - Имплементација корисничког интерфејса који омогућава корисницима једноставно коришћење алата за анализу и визуелизацију статистике.

5.1. Технологије коришћене у изради

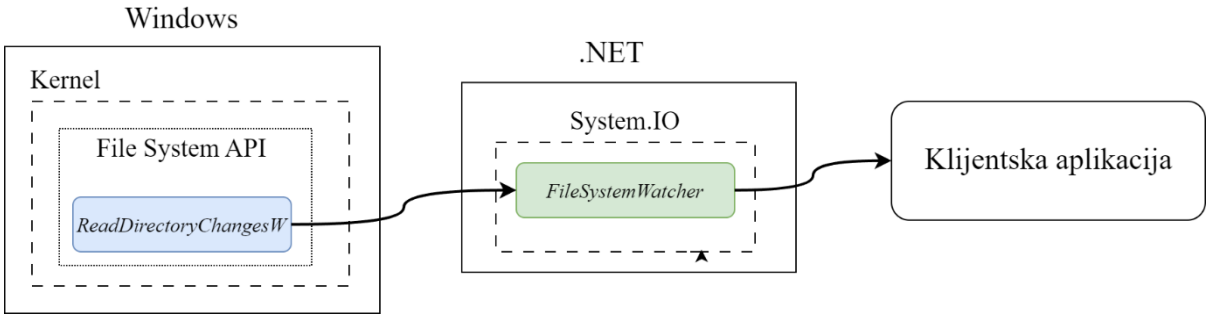
За израду алата за анализу и визуализацију статистике датотека у фајл систему коришћен је програмски језик C# (верзија C# 10.0) у оквиру .NET 6.0 *framework*-а. Пројекат је имплементиран као WinUI 3 десктоп апликација. WinUI 3 је native платформска компонента за израду корисничких интерфејса која се испоручује са Windows App SDK-ом (потпуно одвојена од Windows SDK-а). Windows App SDK пружа уједињени скуп API-ева и алата који се могу користити за креирање десктоп апликација које циљају Windows 10 и новије верзије оперативних система. WinUI 3 представља наследника технологије UWP (Universal Windows Platform). Ове апликације се могу објавити и у Microsoft Store-у.

Алат је намењен за коришћење на Windows оперативним системима, укључујући Windwos 10 или касније верзије.

За обезбеђивање перзистентности података коришћен је SQLite систем за управљање базама података.

За евидентирање информација о раду апликације коришћена је Serilog библиотека која пружа структуирани систем за логовање.

Праћење промена у директоријумима и над датотекама заснива се на имплементацији *FileSystemWatcher* класе у .NET framework-у. Ова класа користи Windows File System API, односно функцију *ReadDirectoryChangesW*. Функција *ReadDirectoryChangesW* прати наведени директоријум и прикупља информације о променама које се дешавају у датотекама и поддиректоријумима. Инфромације о променама се стављају у бафер који апликација може да прочита. Windows опертивни систем користи механизам за обавештавање који је део кернела и који прати систем датотека. Када дође до промена у директоријумима и датотекама, кернел генерише обавештење које функција *ReadDirectoryChangesW* прослеђује апликацији.

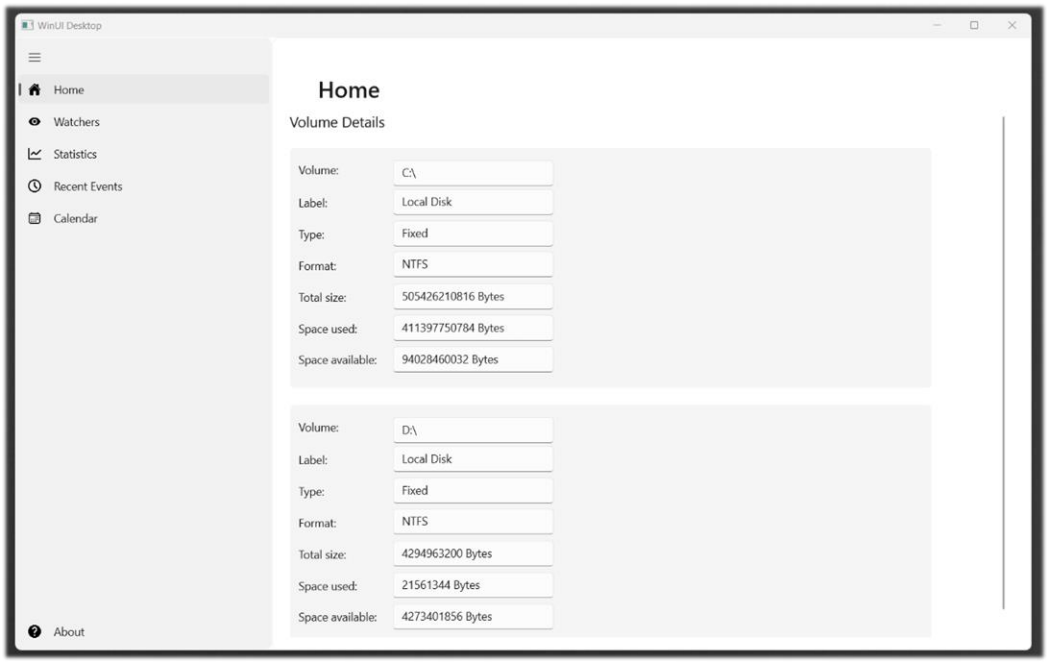


Слика 5.1. – Ток обавештавања корисничке апликације о променама у фајл систему

5.2. Функционалности алата

На почетној страници апликације приказује се листа свих доступних партиција на систему и њихове основне информације:

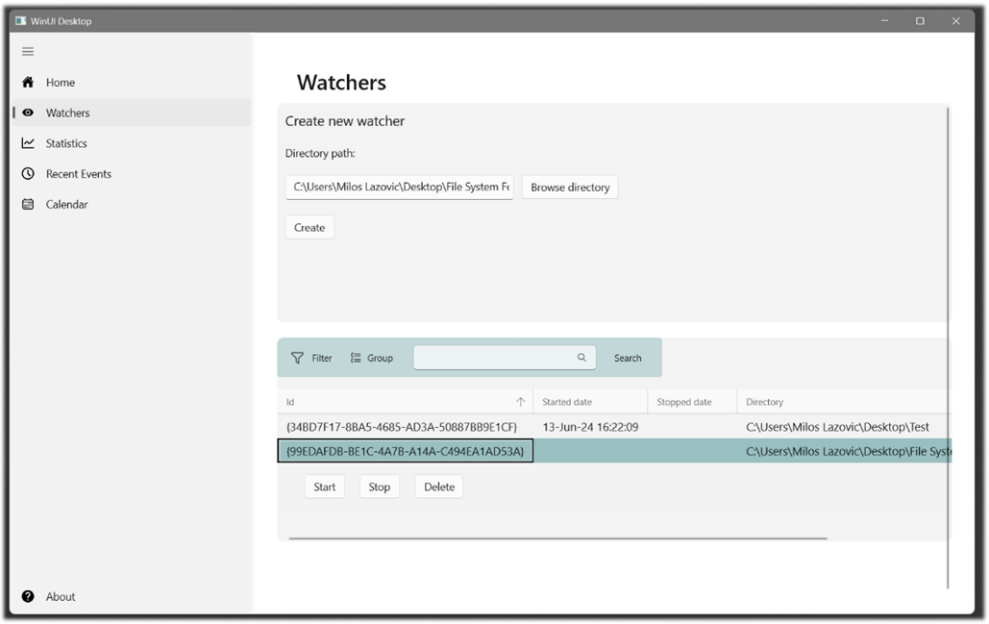
- Назив паритције
- Ознака партиције
- Тип партиције (fixed, removable)
- Формат/тип фајл система
- Укупна величина фајл система у бајтовима
- Заузети простор у бајтовима
- Слободан простор у бајтовима



Слика 5.2. – Изглед Home стране са освновним информацијама о свим партицијама

На другој страници постоји форма за креирање *Watcher*-а навођењем путање до директоријума који желимо да надгледамо. *Watcher* је компнента која обавештава користника о свих догађајима у дефинисаном директоријуму.

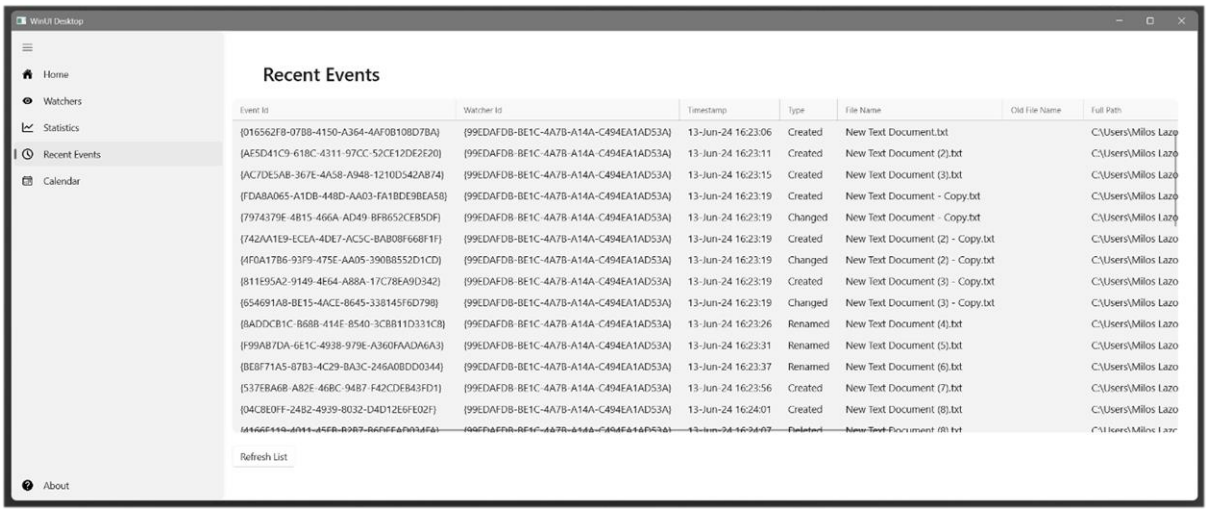
Испод форме за креирање налази се листа свих доступних watcher-а. Селекцијом једног од њих појављују се опције за покретање његове активности, заустављање његове активности и брисање watcher-а.



Слика 5.3. – Страна Watchers за креирање и приказ листе постојећих watcher-а

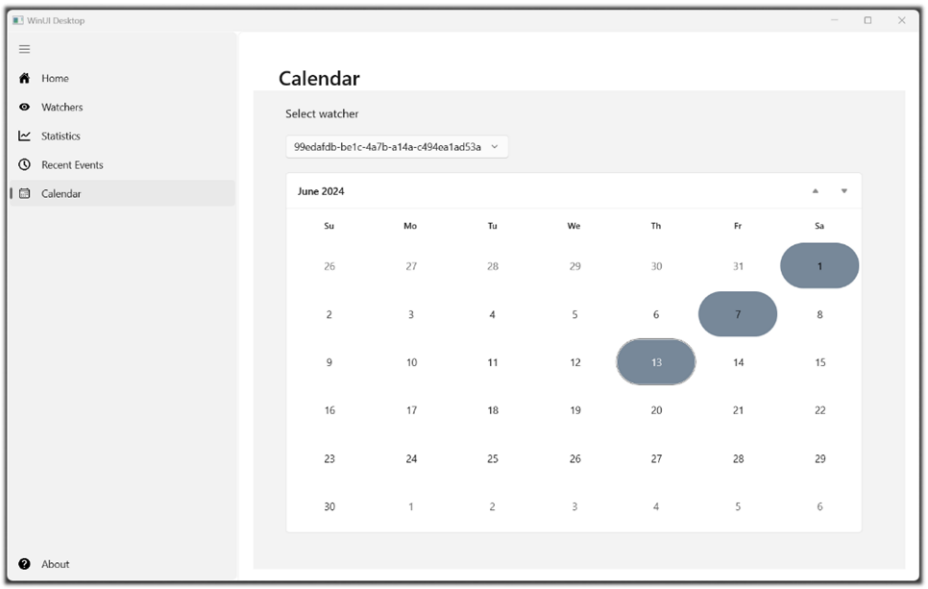
Слика 5.4. – Изглед Statistics стране

На страници *Statistics* приказују се четири графикона која пружају кориснику информације о томе када су биле највеће активности, које промене су се десиле и над којим типом фајлова. Први графикон *Event Types* приказује укупан број свих догађаја одређеног типа који се десио. Други графикон *Most used file types* приказује над којих типом фајлова су се десили ти догађаји. Трећи графикон *Activity in last hour* приказује активност у поседњих сат времена и четврти графикон *Event activity history* приказује којим данима су се дешавали догађаји.

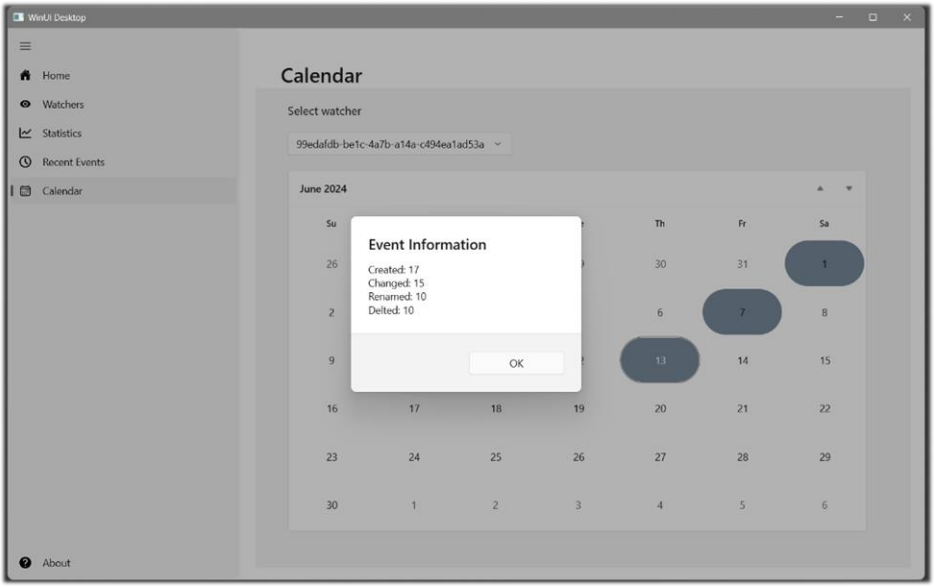


Слика 5.5. – Изглед Recent Events стране

На страници *Recent Events* дат је приказ свих догађаја који су се десили у фајл систему и основне информације о њима.



Слика 5.6. – Изглед Calendar стране

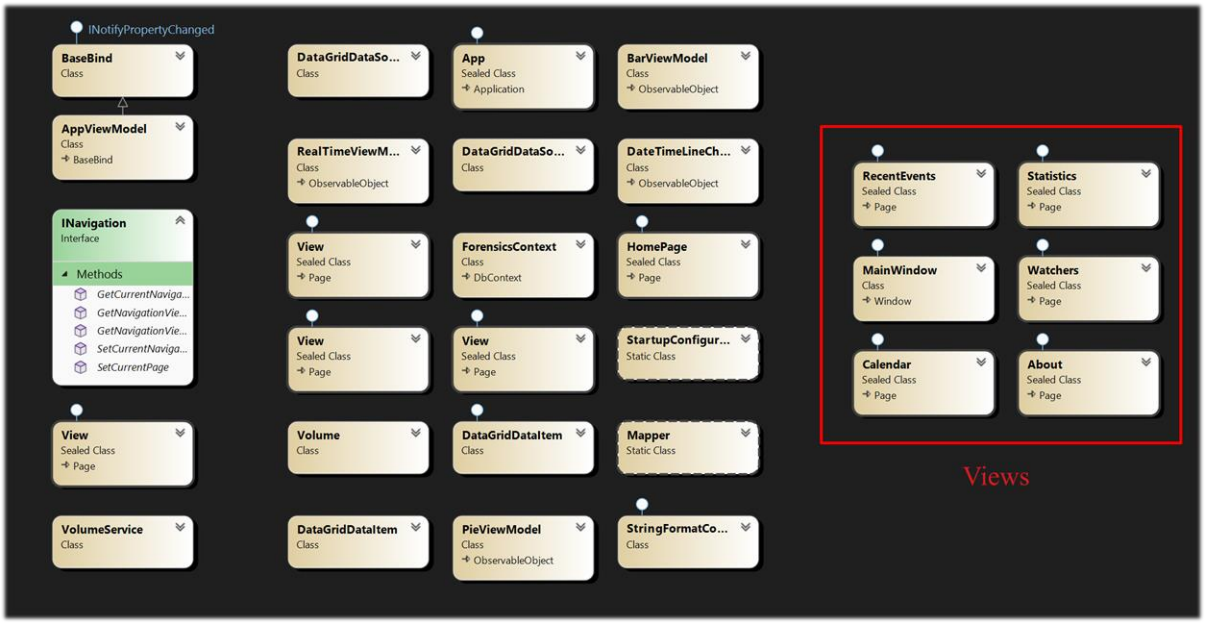


Слика 5.7. – Приказ модалног прозора са основним инфорамцијама о активности за задати дан у години

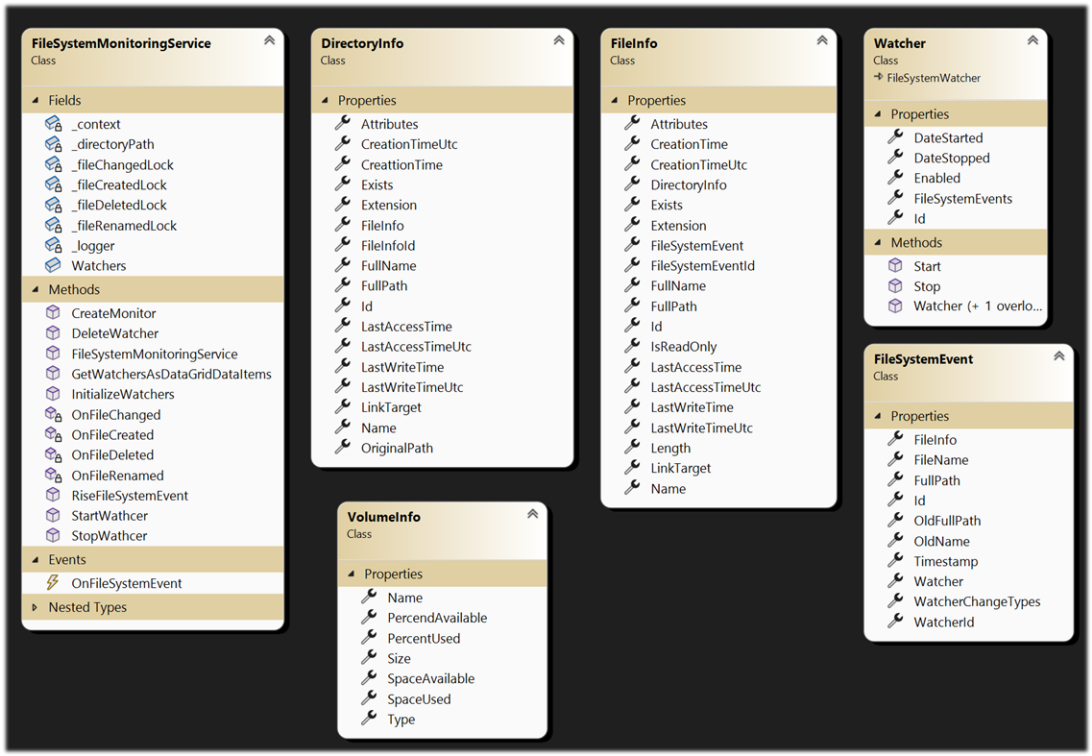
На страници *Calendar* приказан је календарски приказ свих догађаја за изабрани Watcher. Селектовањем неког дана приказује се број догађаја који су се десили за сваки тип: креирање, измена, преименовање, брисање.

5.3. Класни дијаграм

На следећим сликама дати су класни дијаграми класа којима се представља домен проблема и класа које чине основу WinUI 3 апликације.



Слика 5.8. – Дијаграм класа



Слика 5.9. – Дијаграм класа

6. Закључак

У овом семинарском раду истражили смо структуру и принципе рада неколико најзначајнијих фајл система, укључујући NTFS, Ext2, Ext3, UFS1 и UFS2. Разумевање ових фајл система од кључног је значаја за форензичку анализу, јер сваки од њих има специфичне структуре података, механизме за управљање фајловима и метаподацима, као и методе за заштиту и опоравак података.

NTFS је детаљно анализиран због своје сложене структуре и напредних функција као што су journaling и подршка за велики број атрибута повезаних са фајловима. Ext2 и Ext3 фајл системи, као претходници новијих Ext4, такође су обрађени због њихове широко распрострањене употребе у Linux оперативним системима. Посебна пажња посвећена је њиховој организацији у блок групе и начином управљања метаподацима преко inode-а.

UFS1 и UFS2, који су често коришћени у BSD системима, разматрани су у контексту њихове историјске и техничке важности. Ови фајл системи су посебно интересантни због своје структуре цилиндричних група и начина на који чувају резервне копије важних података, што их чини робусним и поузданим.

Циљ овог рада био је да се прикаже како различити фајл системи организују и управљају подацима, и како те разлике утичу на форензичку анализу. Препознавање и разумевање структура података као што су суперблокови, inode-и, и битмапе је кључно за успешну анализу и опоравак података у форензичким истрагама.

Фајл системи играју критичну улогу у управљању подацима, а њихово дубинско познавање је неопходно за ефикасну форензику и очување дигиталних доказа.

7. Референце

[1] File System Forensic Analysis, Brian Carrier, Addison Wesley Professional, (2005)

[2] Incident Response & Computer Forensics Third Edition, Jason Luttgens, Matthew Pepe, Kevin Mandia, Mc Graw Hill Education (2014)

[3] Learn Computer Forensics Second Edition, William Oettinger, Packt, (2022)

[4] NTFS File System and Cloud Computing Forensics, Part III. Advanced Techniques and Tools for Digital Forensics, University of Lisbon

[5] The Art of Memory Forensics, Michael Hale Ligh, Andrew Case, Jamie Levy, Aaron Walters, Wiley (2014)

[6] https://en.wikipedia.org/wiki/Hard_disk_drive

[7] WinUI 3 документација: https://learn.microsoft.com/en-us/windows/apps/winui/winui3/