System V File System

Борисав Живановић

15. март 2023.

Увод

- До сада, били смо упознати са радом са директоријумима и фајловима из перспективе корисника
- Сада желимо да видимо како су ти подаци организовани на диску
  - и како функционишу библиотеке са којима смо до сада приступали фајловима
- Неопходно је да се подсетимо основих појмова из архитектуре рачунара и оперативних система

Шта рачунар заиста зна да ради?

- Језик рачунара: **скуп инструкција** (енгл. ISA, Instruction Set Architecture)
- Аритметичке операције: **add**, **sub**, **div**, **mul**, ...
- Померање података:
  - са улазног уређаја у меморију
  - из меморије на излазни уређај
  - са једне меморијске локације на другу
- Условно гранање: извршавање кода уколико је логички услов испуњен

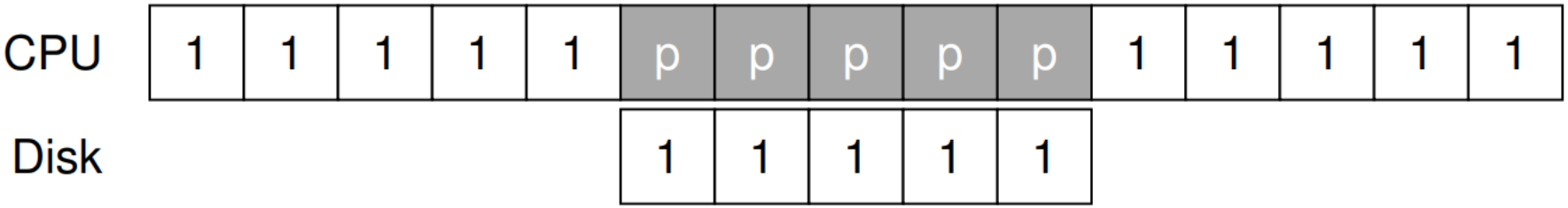
Померање података

- Процесор може да ради искључиво са вредностима које се налазе у радној меморији или регистарима
- Померање података по радној меморији/регистрима је релативно једноставно
  - већина инструкција раде са меморијским адресама и регистрима директно
- Комуникација са улазно/излазним уређајима је нешто компликованија
  - потребно је послати команду уређају
  - прочитати одговор
  - одговор сачувати у радној меморији и даље вршити обраде над њим

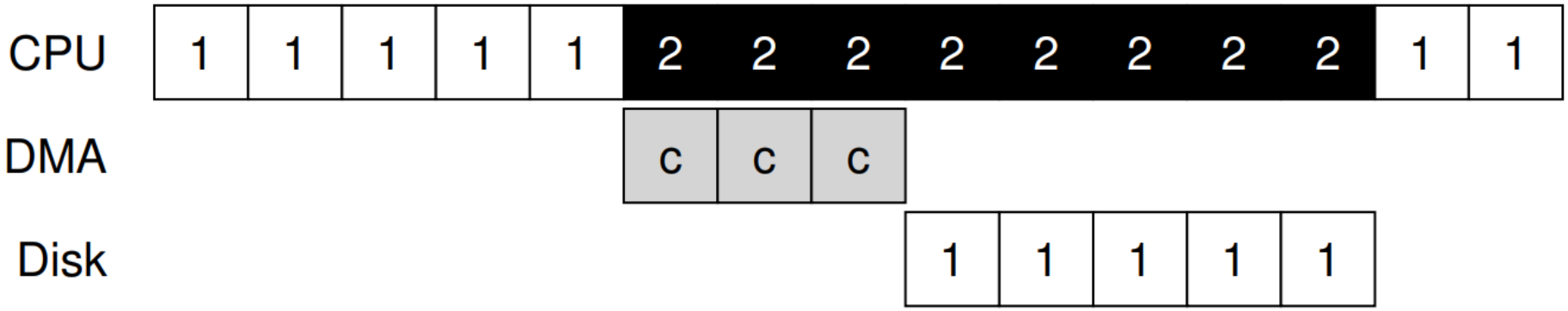
I/O механизми

- **Програмирани I/O**: процесор директно управља уређајима, шаље захтев и чека одговор
  - **Memory Mapped I/O**: уређаји су представљени меморијским адресама, користимо обичне инструкције
  - **Port Mapped I/O**: уређаји су представљени бројем порта, користимо посебне инструкције (x86: IN, OUT)
- **Direct Memory Access (DMA)**: DMA контролер директно приступа радној меморији и прекидом обавештава процесор о извршеној акцији
  - подаци се чувају у радној меморији
  - минимална комуникација са DMA контролером преко програмираног I/O

Програмирани I/O



Direct Memory Access (DMA)



Меморијска хијерархија I

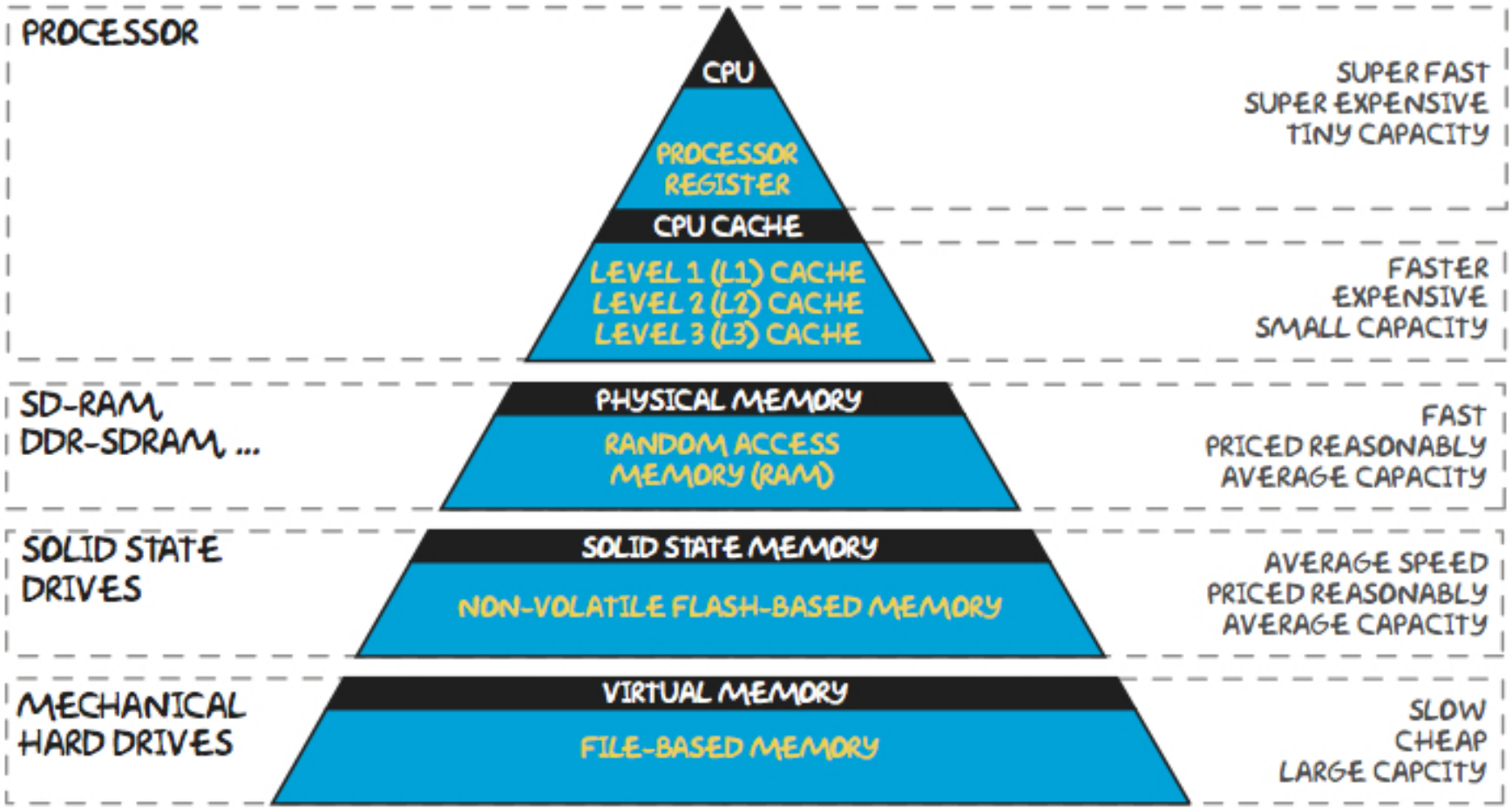
Ideally one would desire an indefinitely large memory capacity such that any particular... word would be immediately available... We are... forced to recognize the possibility of constructing a hierarchy of memories each of which has greater capacity than the preceding but which is less quickly accessible.

Burks, Goldstine, von Neumann (1946)

Меморијска хијерархија II

- Проблем: не постоји бесконачно брза и бесконачно велика меморија
- Чињеница: постоје технологије меморије које омогућавају релативно велики капацитет, по цену релативно мале брзине
  - ...као и обрнуто!
  - брзина и капацитет меморије су, по правилу, обрнуто сразмерни
- Да ли је могуће добити највећи капацитет уз највећу брзину, по најмањој цени?
- Меморијска хијерархија нам ово *донекле* омогућава
  - цена: *приближно* најспорија меморија
  - брзина: *приближно* најбржа меморија

Меморијска хијерархија III



Контрола приступа у хардверу

- Рачунар без контроле приступа би донекле био употребљив у једнокорисничком окружењу
  - ...али неупотребљив у вишекорисничком
  - чак и у једнокорисничком окружењу, одсуство изолације процеса представља велику опасност
- Основне градивне блокове је неопходно имплементирати у хардверу
  - софтвер можда неће бити рад да сарађује!
- Кључни механизми: режими рада процесора, виртуелна меморија

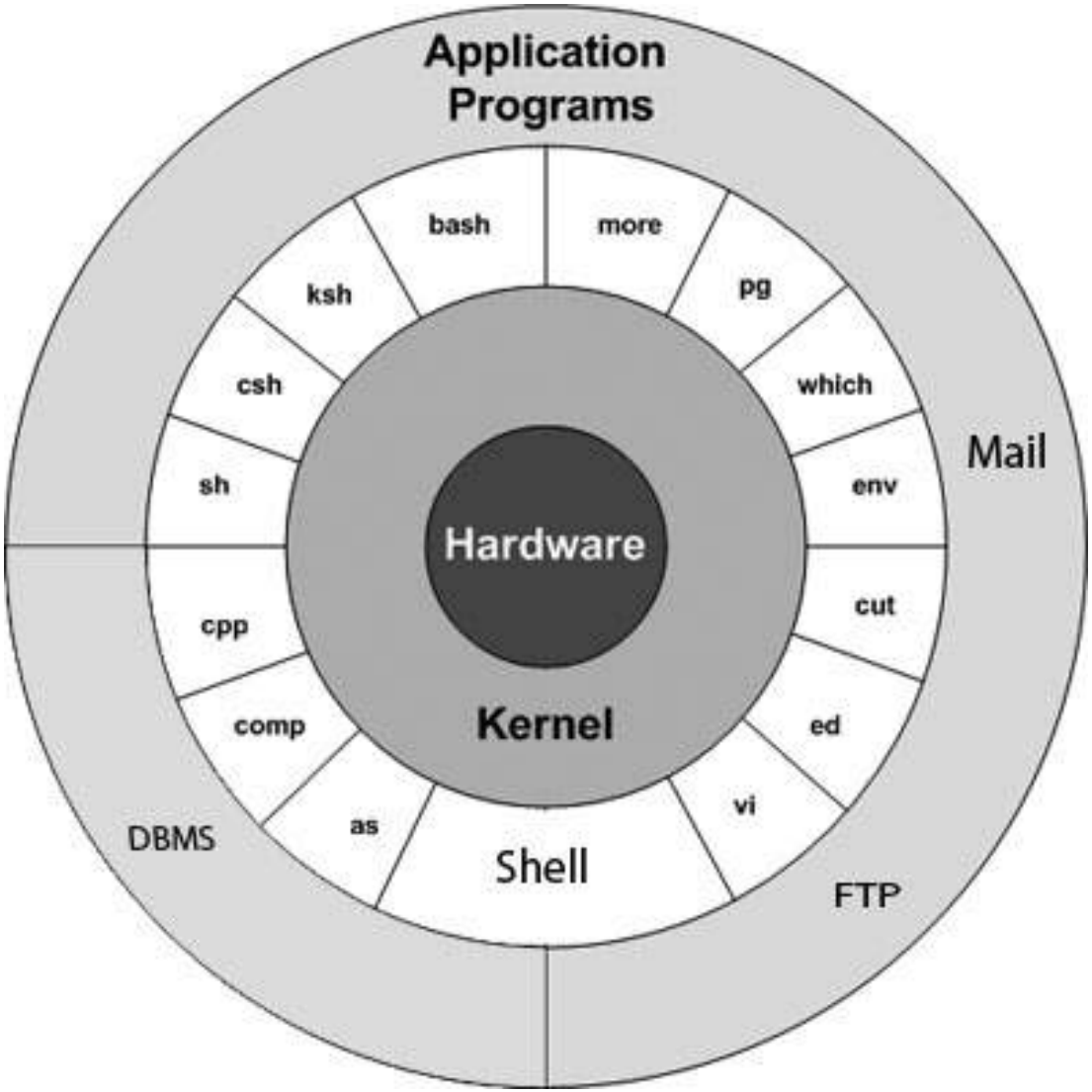
Режими рада процесора

- Привилеговани: IO, меморијске табеле, табеле прекида
  - кернел
- Неривилеговани: аритметичко/логичке операције, условно гранање, ограничен приступ меморији, системски позив
  - кориснички софтвер
- Прелазак из непривилегованог у привилеговани режим је могућ приликом прекида или системског позива
- Кернел одбија захтев уколико кориснички процес нема потребне привилегије и убија га

Покретање оперативног система I

- Процесор се буди у привилегованом режиму
- Учитава се кернел
- Иницијализују се табеле прекида
- Иницијализују се меморијске табеле
- Контрола се предаје корисничким програмима, прелази се у непривилегован режим
- Овако подешен посредник (кернел) више није могуће уклонити или заобићи
  - ...под претпоставком да нема багова у имплементацији кернела и хардвера

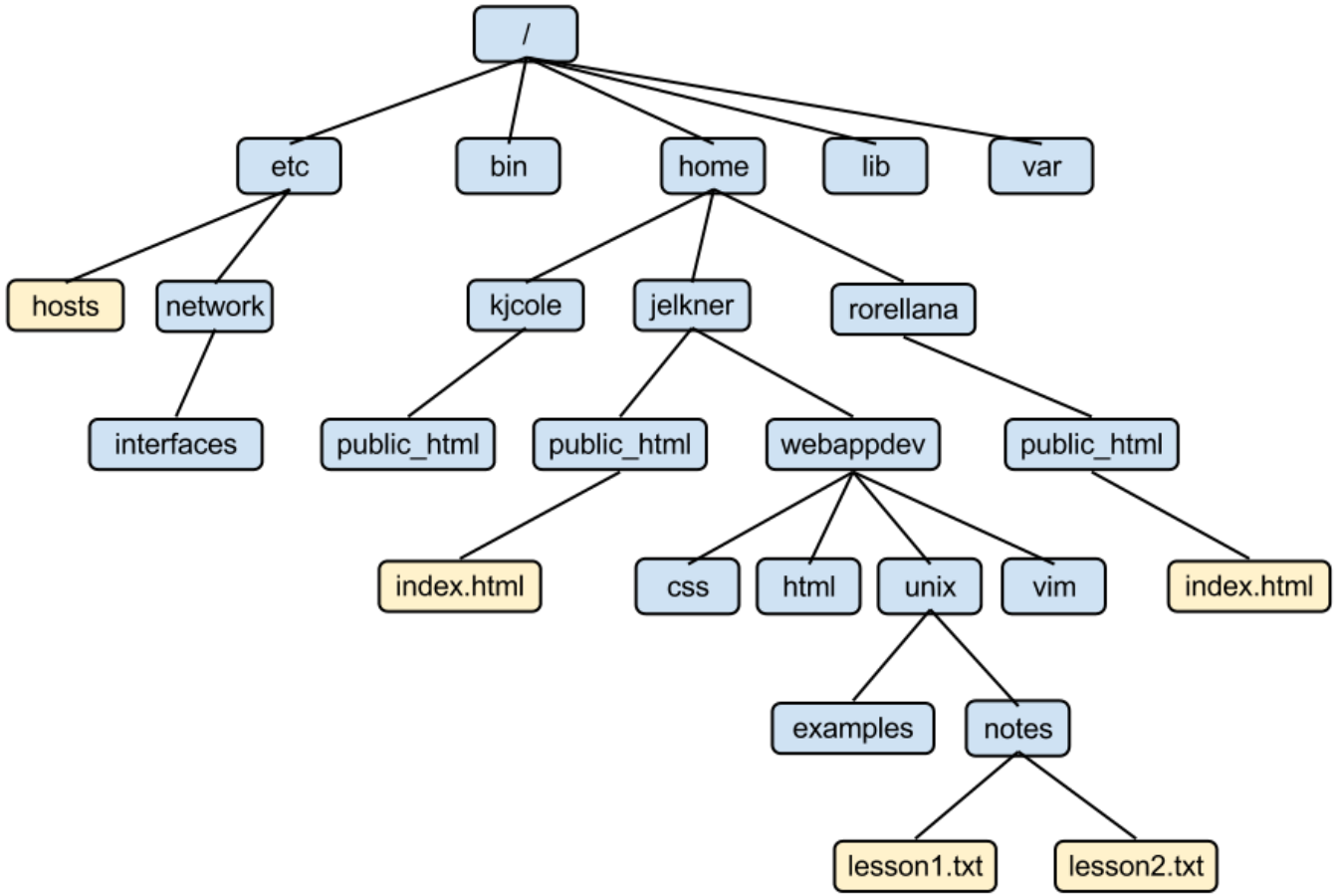
Покретање оперативног система II



Фајл систем I

- Представља формат организације података на трајном складишту
- Хијерархијска структура (стабло)
- Уводи две главне апстракције:
  - **фајл**: именовани запис
  - **директоријум**: групише друге директоријуме и фајлове
- Додатно: контрола приступа фајловима
  - IO инструкције се извршавају у привилегованом режиму
  - интеракција са трајним складиштем је могућа искључиво преко системских позива
  - систем одбија извршавање акције уколико корисник нема овлашћење

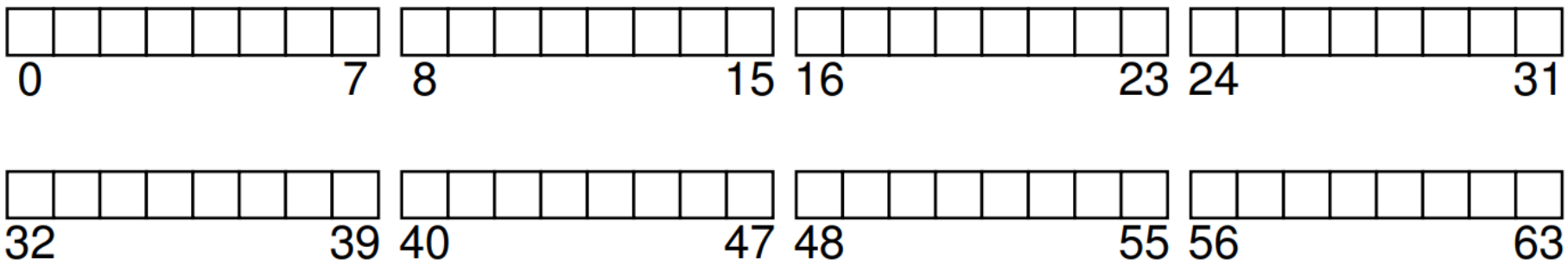
Фајл систем II



UNIX file API

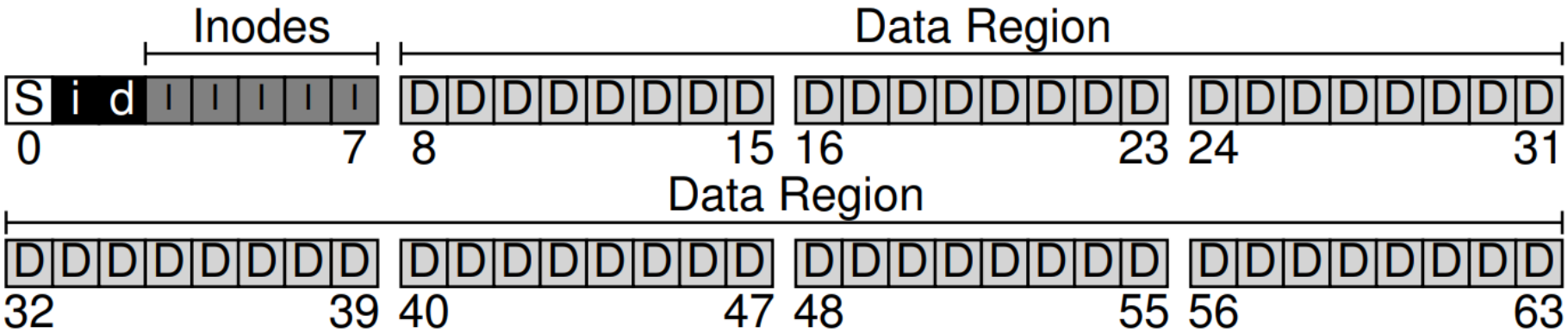
- **FILE *fopen(const char *filename, const char *mode)**
  - отварање фајла на задатој путањи
- **size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)**
  - читање отвореног фајла
- **size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)**
  - испис података у отворени фајл
- **int fseek(FILE *stream, long int offset, int whence)**
  - померање показивача у отвореном фајлу
- **int fclose(FILE *stream)**
  - затварање фајла

Трајно складиште



Трајно складиште можемо посматрати као **адресни простор**. Разлика у односу на радну меморију је да је најмања јединица коју је могуће адресирати 512 бајтова.

Структура фајл система



Superblock: метаподаци о фајл систему
Inodes: метаподаци о садржају фајл система, један inode представља један фајл или директоријум
Data Region: блокови који представљају садржај датотека

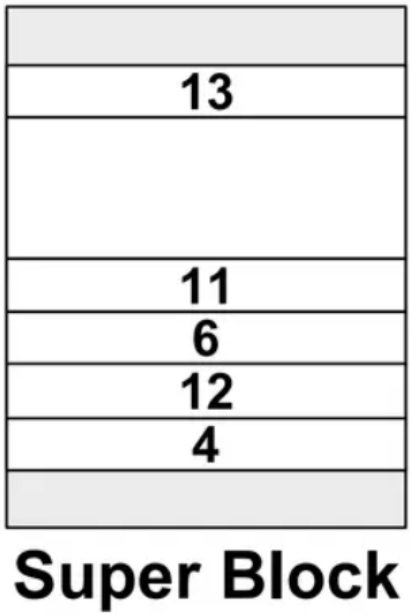
Superblock

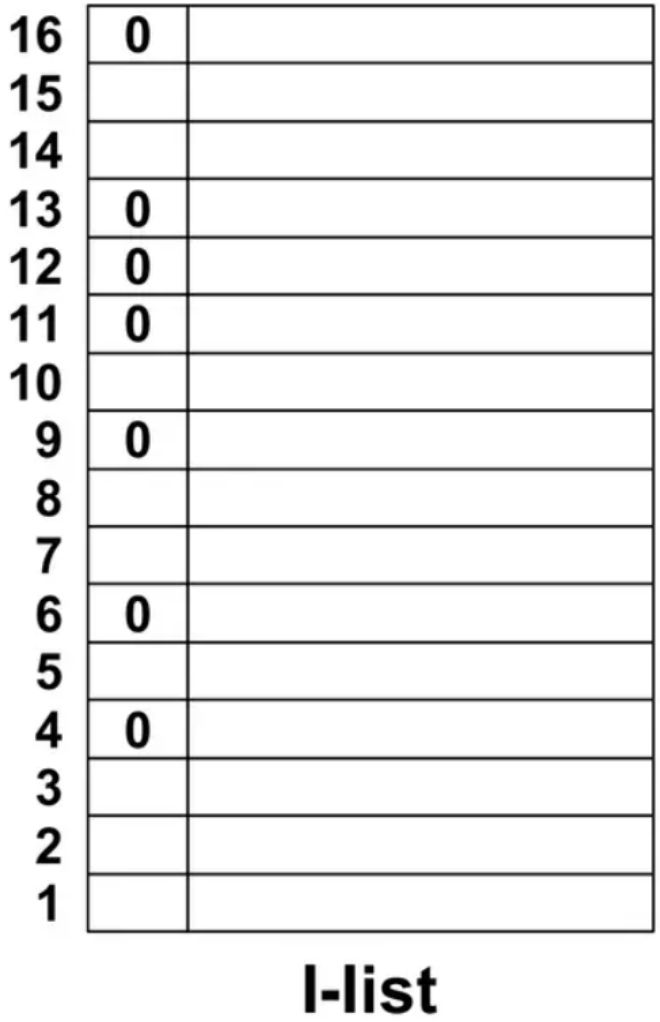
- Када покушавамо да интерпретирамо садржај фајл система, неопходно је да имамо полазну тачку
- Садржи основне податке о фајл систему
  - величину фајл система
  - величину листе inode-ова
  - број слободних inode-ова и data block-ова
  - први део листе слободних data block-ова
  - неке слободне inode-ове (налик кешу)

Free inode list I

- Неопходно је да знамо који inode-ови су слободни, како би могли да их искористимо за креирање фајлова/директоријума
- У inode листи, слободни inode-ови су означени тако што је $mode = 0$
- Додатну (кеширану) листу садржи superblock

Free inode list II

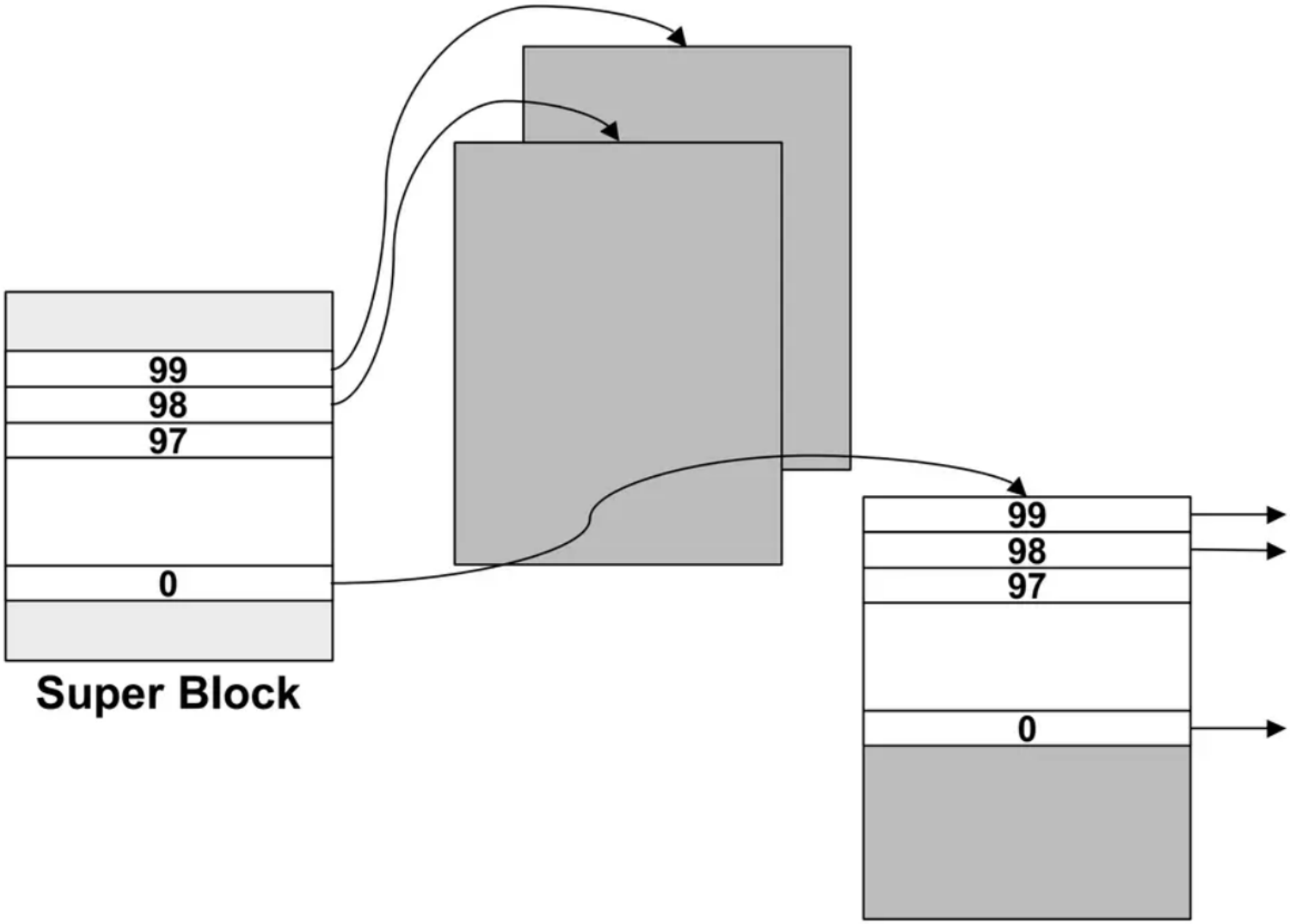




Free data block list I

- Неопходно је да знамо који data block-ови су слободни, како би могли да их искористимо за чување садржаја фајлова/директоријума
- Први део (100 показивача) се налази у superblock-у
  - уколико то није довољно да би се описао слободан простор, задњи показивач показује на листу од још 100 слободних блокова
  - ...и тако даље, док сав слободан простор није описан
- Структура налик на комбинацију array list и linked list
- Ослобађање data block-а изазива реконструкцију лист

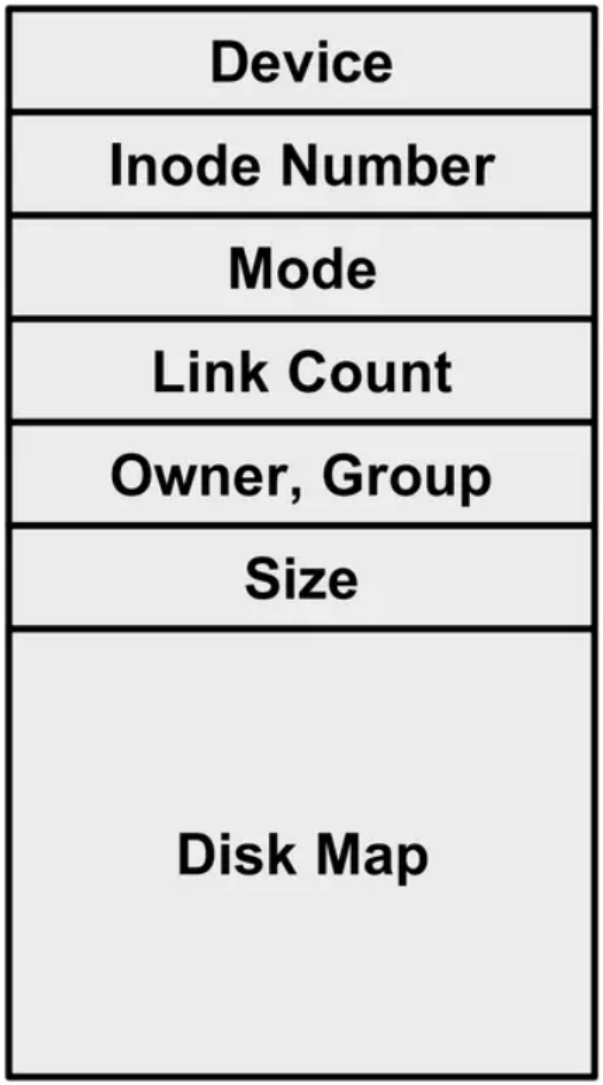
Free data block list II



Inode I

- Један inode (index node) представља један фајл или директоријум
- Уколико представљамо фајл, inode показује на блокове који чувају садржај
- Уколико представљамо директоријум, inode представља фајл са посебном стурктуром
  - у суштини, парови $(filename, inodeID)$
  - и даље важе сва правила чувања обичних фајлова!
- Број inode-ова ограничава број фајлова/директоријума које је могуће представити фајл системом
- Додатно: подаци о власништву (због контроле приступа)

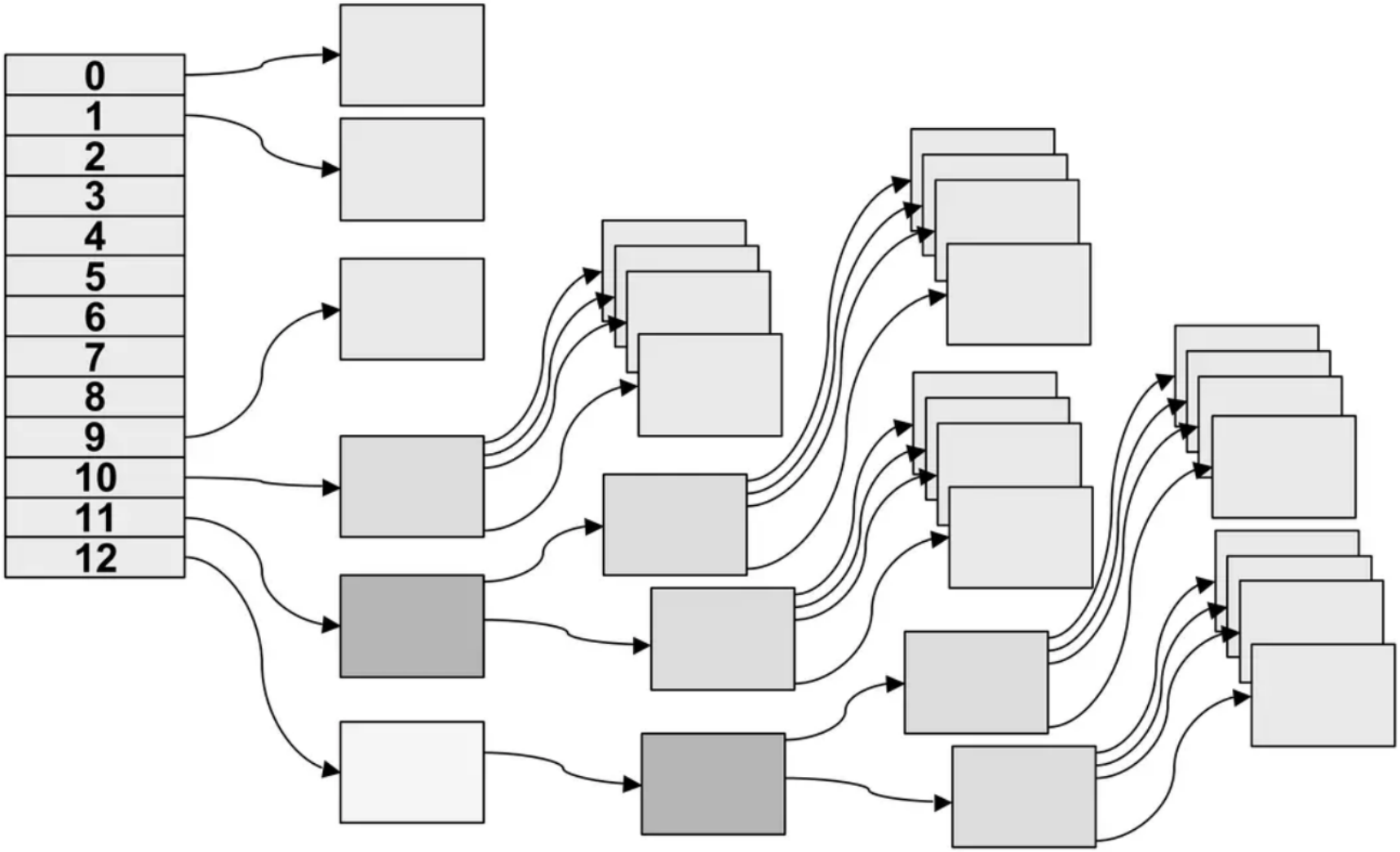
Inode II


Device
Inode Number
Mode
Link Count
Owner, Group
Size
Disk Map

Inode Disk Map I

- Представља асиметрично стабло
- Показивачи 0-9 су директни
  - чувају адресу data block-а који садржи податке
- Показивач 10 је индиректан
  - чува адресу data block-а који садржи адресе других data block-ова
- Показивач 11 је двоструко индиректан
- Показивач 12 је троструко индиректан
- Нивои индирекције су уведени због односа величине метаподатака и садржаја фајлова
- Број показивача ограничава величину фајла који је могуће представити у фајл систему

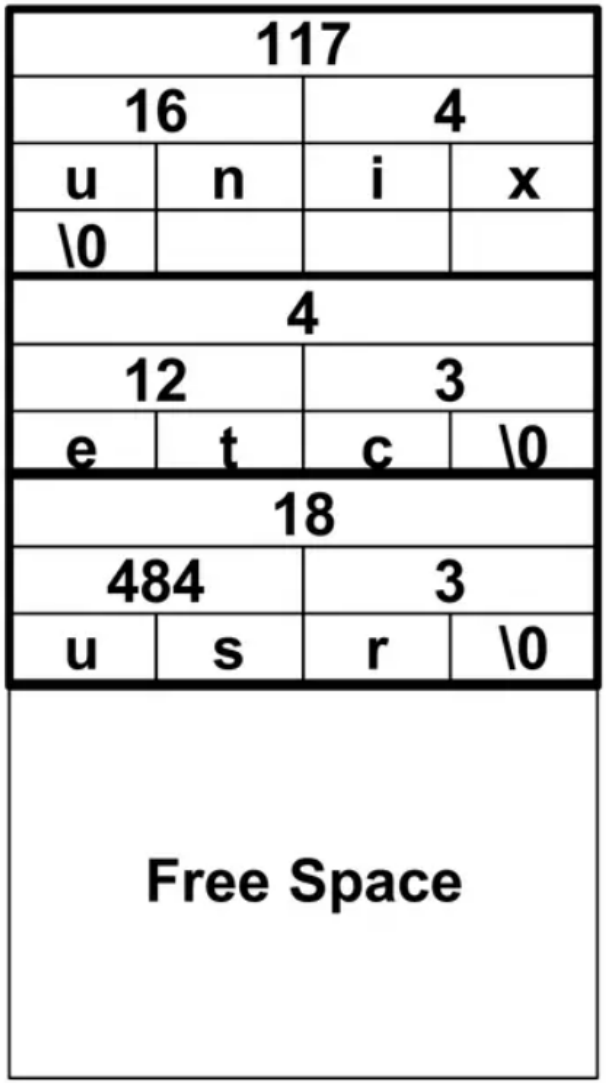
Inode Disk Map II



Формат директоријума I

```
inum | reclen | strlen | name
   5       12        2     .
   2       12        3     ..
  12       12        4     foo
  13       12        4     bar
  24       36       28     foobar_is_a_pretty_longname
```

Формат директоријума II



Читање фајла I

- Желимо да прочитамо фајл /**foo**/**bar**
- Да би добавили садржај, потребно је наћи inode који представља **bar**
- Путању растављамо на делове [/, **foo**, **bar**]
- Проналазимо inode који представља /
  - то је увек inode ID 2, проналазимо га у листи по индексу
- Читамо садржај директоријума /, проналазимо inode који представља **bar**
- Читамо садржај директоријума **bar**, проналазимо inode који представља **foo**
- Читамо садржај фајла **foo**
- Уписује се време прступа у inode **foo**

Читање фајла II

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data [0]	bar data [1]	bar data [2]
open(bar)			**read**			**read**				
				read			**read**			
					read					
read()					**read**			**read**		
					write					
read()					**read**				**read**	
					write					
read()					**read**					**read**
					write					

Исписивање фајла I

- Желимо да испишемо нови фајл **/foo/bar**
- Проналазимо слободан inode
- Креирамо нови inode који ће да представља **bar**
- inode **bar** означавамо као заузет
- Проналазимо директоријум **/foo**
- У фајл који представља директоријум додајемо нови запис $(foo, newID)$

Исписивање фајла II

	data bitmap	inode bitmap	root inode	foo inode	bar inode	root data	foo data	bar data [0]	bar data [1]	bar data [2]
create (/foo/bar)		**read** **write**	**read**	**read**	**read** **write** **write**	**read**	**read** **write**			
write()	**read** **write**				**read** **write**			**write**		
write()	**read** **write**				**read** **write**				**write**	
write()	**read** **write**				**read** **write**					**write**

Питања

- На које начине можемо да избришемо фајл?
- Због чега је команда **rm** брза без обзира на величину фајла?
- Који кораци би били неопходни за опоравак фајла?
- На које начине можемо да преместимо фајл/директоријум са једне путање на другу?
- Због чега је **mv** брз, а copy/paste спор?

Литература

- Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau & Andrea C. Arpaci-Dusseau
- File Systems, Thomas Doeppner
- Unix File System, Sechang (Sonny) Son
- Computer Organization and Design: The Hardware/Software Interface, David A. Patterson & John L. Hennessy
- Системски софтвер (презентације), Иван Нејгебауер
- Operating Systems: Internals and Design Principles, William Stallings
- Preliminary Discussion of the Logical Design of an Electronic Computing Instrument