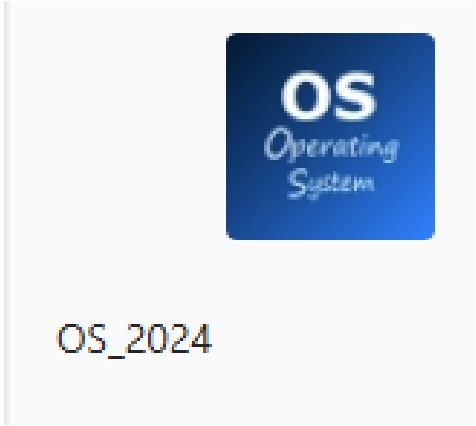
Teams - **93mi85a**



Управление на паметта

random-access memory

Деклариране на променливи

- short years = 2000;
- int days = 730480;
- long hours = 17531520;

Съдържание

- Статично разпределение
- Преместване на програми/процеси
- Припокриване (overlay)
- Размяна (Swapping) (приоритети,времеделене)
- Основни функции на управлението
- Разпределяне на непрекъсната област памет (ос/потребител)
- Разпределяне на рздели
- Фиксирани раздели – вътрешна фрагментация
- Променливи раздели –външна фрагментация
- Премествaеми раздели
- Базови регистри
- Разпределяне на константи, данни и променливи в отделни дялове
- Стратегии най-подходящ, първи подходящ, стратегия на близнаците...
- Страници – таблици на страниците, таблици на кадри...
- Сегменти
- Защита на кадри –флаг за достъп
- Разделяне на сегменти на страници

Основи на управлението от ОС

- Информация за заето/свободно пространство;
- Стратегии за разпределение;
- Заделяне на памет;
- Освобождаване на памет;

Направления

- Основни изисквания към управлението на паметта
- Разделяне на паметта на дялове
- Основни блокове при управлението на паметта
  - Разделяне на страници
  - Сегментиране

Необходимостта от управление на паметта

- Паметта днес е евтина и става все по-евтина но приложенията изискват все повече и повече памет и тя никога не стига!
- Управлението на паметта включва размяна на блокове данни с вторичното хранилище
- Входно-изходните операции с паметта са бавни в сравнение с тези при процесора
- Операционната система трябва умно да избере момента за размяна, за да се максимизира ефективността на процесора

Управление на паметта

Паметта трябва да се разпредели, за да се гарантира, че готовите процеси ще имат на разположение достатъчно процесорно време

Изисквания към управлението на паметта

- Преразпределяне (Relocation)
- Защита
- Подялба
- Логическа организация
- Физическа организация

Изисквания: Преместване

- Програмистът не знае, къде в паметта ще отиде програмата, когато се изпълни,
  - Програмата може да бъде прехвърлена на диска и да се върне в основната памет на друго място (преместване) -Swapping
- Указателите в паметта трябва да се преведат като действителните физически адреси в паметта

Управление на паметта - Терминология

Таблица 7.1 Управление на паметта - Терминология

Термин	Описание
Кадър (Frame)	Блок от основната памет, дължината на който е **фиксирана**.
Страница (Page)	Блок с данни във вторичната памет (например на диска), дължината на който е фиксирана.
Сегмент (Segment)	Блок с данни, намиращ се във вторичната памет, дължината на който е променлива.

Изисквания: Защита

- Процесите не трябва да могат да указват места в паметта, използвани от друг процес, без да имат изрично разрешение.
- Не е възможно в момента на компилиране да се проверяват абсолютни адреси
- Трябва да се проверяват при изпълнение
- Таблици за защита на кадрите

Изисквания към ОС : Поделяне

- Да позволява на няколко процеса достъп до една и съща част от паметта
- Възможност за (overlay) припокриване
-

Изисквания : Логическа организация

- Паметта е организирана линейно (обикновено)
- Програмите се пишат като модули
  - Модулите могат да се пишат и компилират независимо един от друг
- Различни степени на защита за различните модули (права само за четене, права само за изпълнение)
- Подялба на модулите от процесите на ОС
- Сегментирането особено помага в тези случаи

Изисквания: Физическа организация

- Не може програмистът да се натовари с отговорността да управлява паметта ?!
- Паметта, която е отредена на програмата и данните, може да е недостатъчна.
  - Припокриването позволява на различните модули да бъде отредена една и съща област от паметта, но това отнема време на програмата
- Програмистът не знае с колко място ще разполага

Видове разделяне на дялове

- Фиксирано разделяне на дялове
- Динамично разделяне на дялове
- Просто страницириране
- Просто сегментиране
- Страницириране на виртуалната памет
- Сегментиране на виртуалната памет

Фиксирано разделяне на дялове

- **Дялове с еднакъв размер (виж Фиг. 7.3а)**
  - **Всеки процес, чийто размер е по-малък от или равен на размера на дела може да бъде зареден в наличен дял**
- **Операционната система може да размени процес, който е в дела**
  - **Ако няма процес в готово състояние или в състояние на изпълнение**
  - **(спомняте си основните състояния на процесите ?!?)**

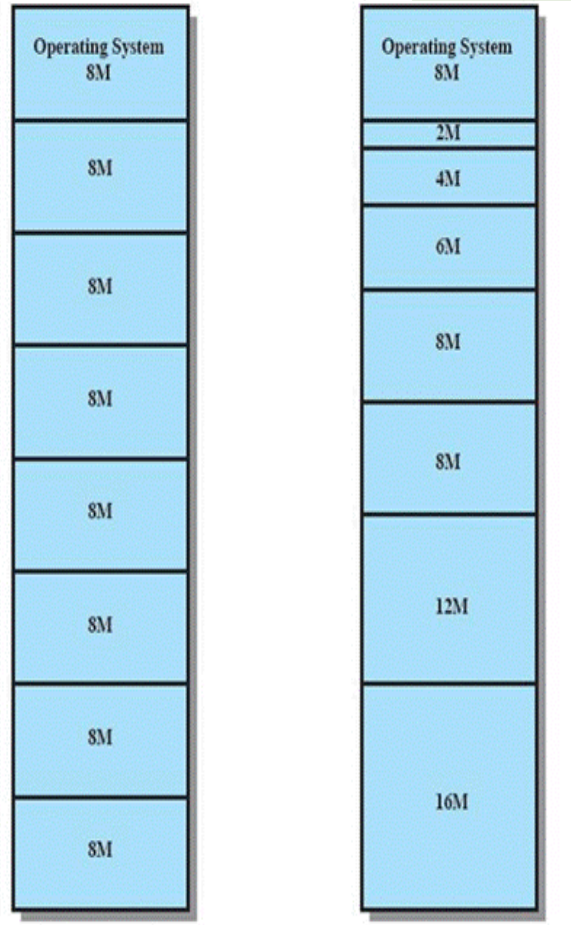


Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Проблеми при фиксираното разделяне на дялове

- Дадена програма може да не се събира в дяла.
  - Програмистът трябва да пренапише програмата, като използва припокриване, но как ?!?.
- Неефективно използване на основната памет.
  - Всяка програма, независимо колко е малка, заема цял дял.
  - Това води до ***<u>вътрешно фрагментиране.</u>***

Решение – дялове с различен размер

- Намалява и двата проблема, но не ги решава напълно
- На Фиг. 7.3b (примерене размер)
  - Програми до 16M се събират без припокриване;
  - По-малките програми могат да се поставят в по-малки дялове, като така се намалява **вътрешното фрагментиране**;

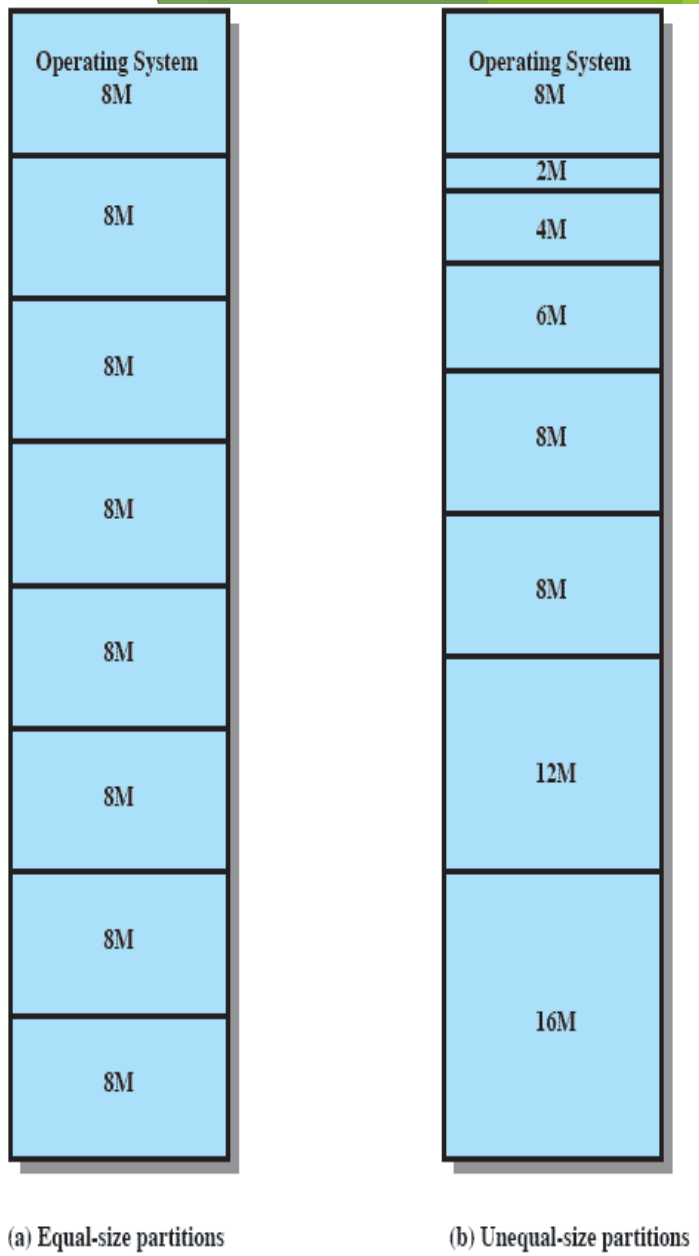


Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Алгоритъм за разполагане

- При равен размер
  - Разполагането е тривиално (има само една възможност)
- При различен размер
  - Всеки процес може да бъде разположен в най-малкия дял, в който се побира
  - Опашка за всеки дял
  - Процесите са разположени по начин, по който се свежда до минимум загубата на място в отделните дялове

Фиксирано разделяне на дялове

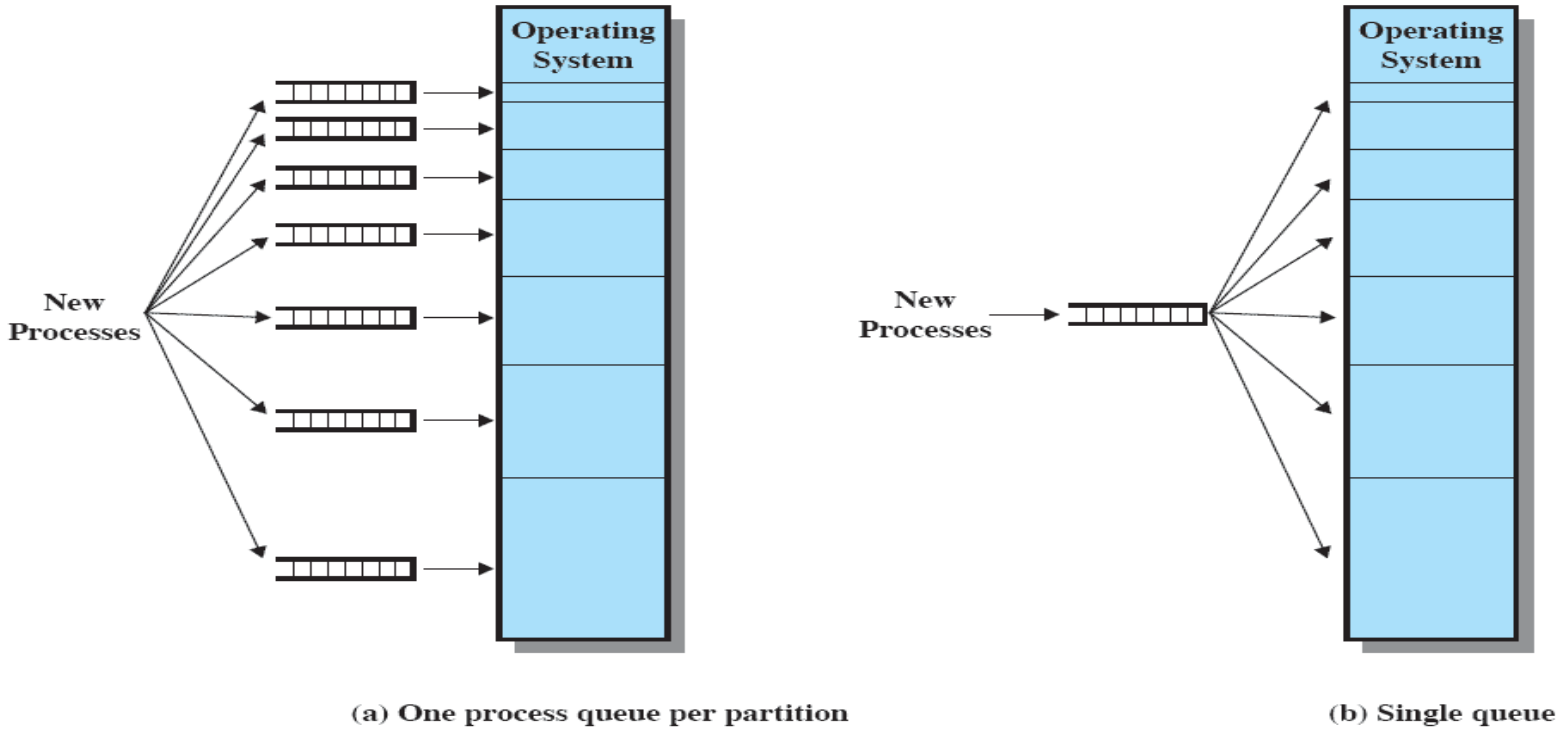


Figure 7.3 Memory Assignment for Fixed Partitioning

а) Опашка от по един процес за дял
б) Една опашка за всички процеси

Фигура 7.3. Разпределение на паметта при фиксирано разделяне на дялове

Проблеми при фиксираните дялове

- Броят на активните процеси е ограничен от системата. Защо ?
  - Ограничен от броя на предварително зададените дялове
- Голям брой много малки процеси няма да използва ефективно мястото

Динамично разделяне на дялове

- Дяловете са с променлива дължина и брой ;
- Всеки процес получава точно толкова памет, колкото има нужда ;

Пример за динамично разделяне на дялове

- ***Външно фрагментиране***
- Памет, която е външна за всички процеси, се фрагментира
- Проблемът може да се реши чрез ,***компактност‘***
  - ОС премества процесите, така че да образуват последователен ред
  - Отнема време и ресурси на процесора

Виж Фигура 7.4

Динамично разделяне на дялове

- Операционната система трябва да реши кой свободен блок да бъде отреден за даден процес
- Алгоритъм за най-добро съответствие (Best-fit algorithm)
  - Избира блока, чийто размер е най-близък до размера на заявеното пространство
  - Като цяло производителността е най-ниска
  - След като най-малкият блок е намерен, остава най-малкото възможно фрагментиране
  - По-често трябва да се прави компактност на паметта

Динамично разделяне на дялове

- Първи подходящ (First-fit algorithm)
  - Сканира паметта от началото и избира първия достатъчно голям свободен блок;
  - Най-бързият метод;
  - В предната част на паметта може да има заредени много процеси, които всеки път трябва да се обхождат отново, докато се намери свободен блок ;

Динамично разделяне на дялове

- Следващ подходящ (Next-fit)
  - Сканира паметта от мястото, където е последното разполагане
  - По-често отрежда блокове към края на паметта, където намира най-големия блок
  - Най-големият блок от паметта се разбива на по-малки блокове
  - Изисква се компактност, за да се намери голям блок в края на паметта

Разпределение

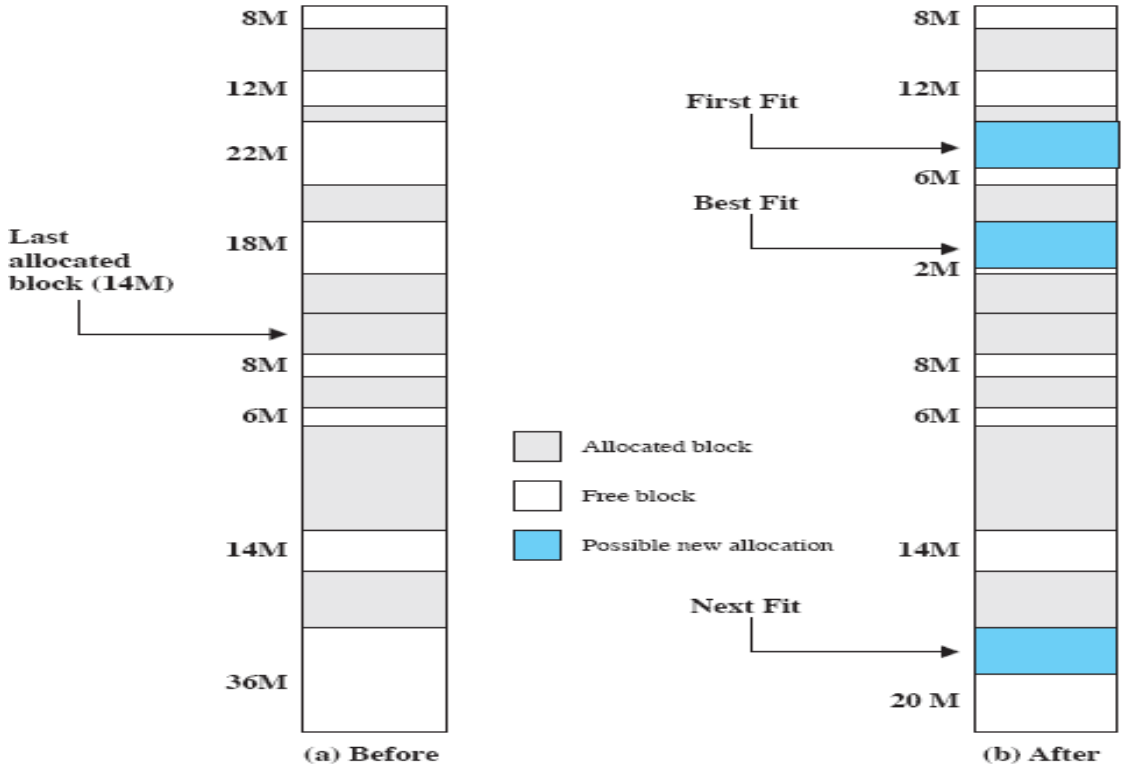


Figure 7.5 Example Memory Configuration before and after Allocation of 16-Mbyte Block

а: Последно разпределен блок,

б: Първи подходящ, Най-подходящ, Зает блок, свободен блок, възможност за ново разпределение

Следващ подходящ Фигура 7.5. Примерна конфигурация на паметта преди и след разпределяне на 16MB блок

Система на близнаците (Buddy System)

- Цялото налично място се третира като един блок с размер 2U
- Ако има заявка с размер *s, където* $2U-1 < s <= 2U$
  - целият блок се заделя
- В противен случай блокът се разделя на 2 равни части
  - Процесът продължава, докато и най-малкият блок по-голям или равен на *s се създаде*

Пример за система на близнаците

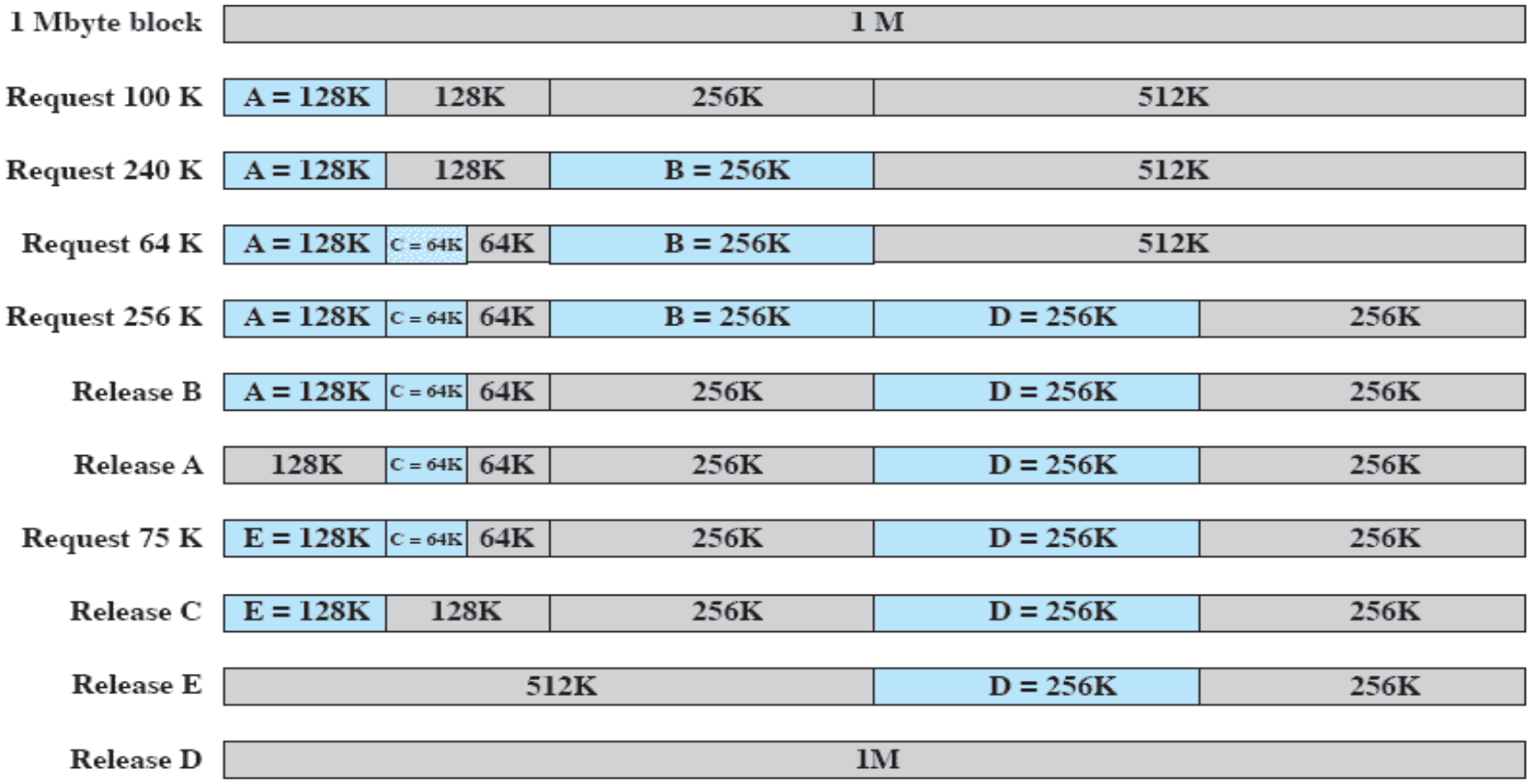


Figure 7.6 Example of Buddy System

Дървовидно представяне на системата на близнаците

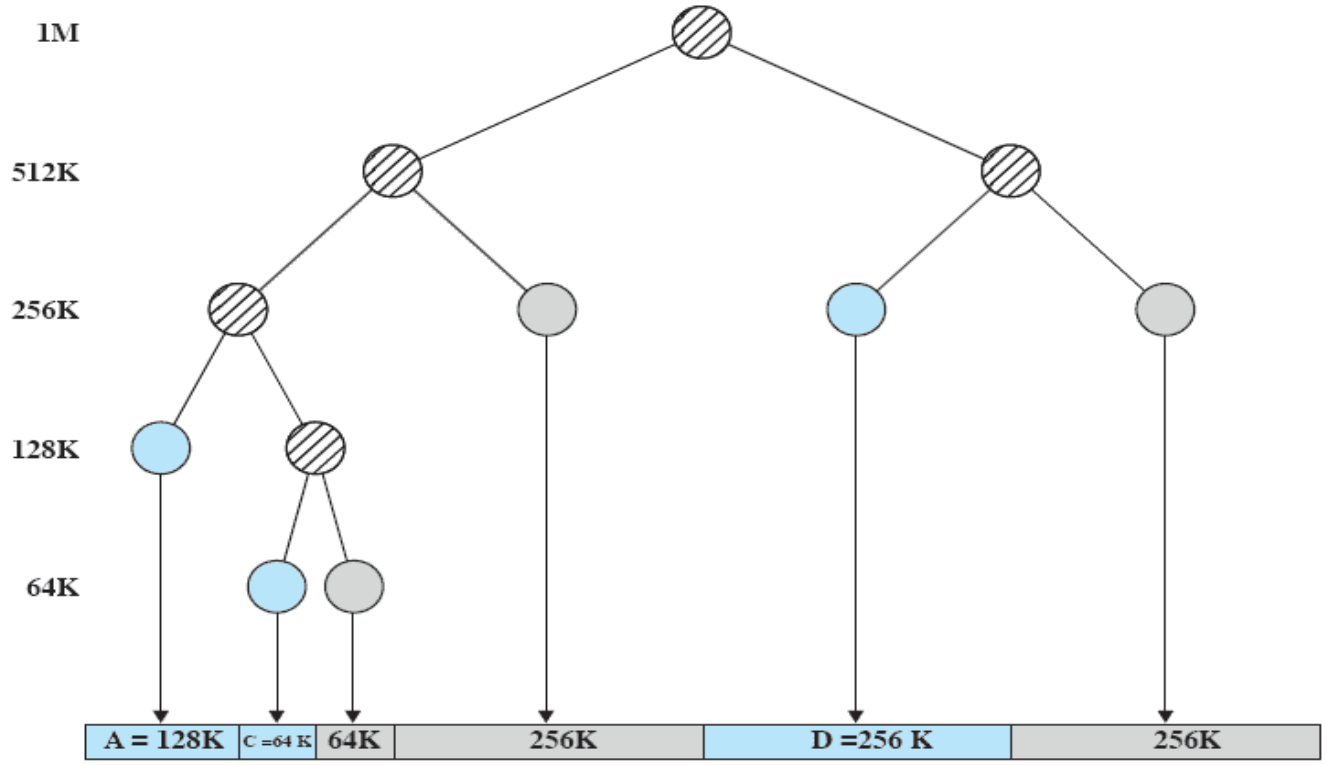


Figure 7.7 Tree Representation of Buddy System

Преразпределяне (Relocation)

- Когато програмата се зарежда в паметта, се определя действителния (абсолютен) размер на местата в паметта
- Един процес може да заема няколко дяла, което означава, че той ще има различни абсолютни места в паметта по време на изпълнение
  - Размяна
  - Compact

Адреси

- Логически
  - Указател към място в паметта, което е независимо от текущото разпределение на данни в паметта.
- Относителни
  - Адрес, който е изразен спрямо някаква известна точка.
- Физически или абсолютни
  - Абсолютният адрес или действителното местоположение в основната памет.

Регистри, използвани по време на изпълнение

- Базов регистър (Base register)
  - Началният адрес за процеса
- Краен регистър (Bounds register)
  - Крайното местоположение на процеса

- Тези стойности са задават, когато процесът се зарежда или когато процесът се разменя

Регистри, използвани по време на изпълнение

- Стойността на базовия регистър се добавя към относителен адрес и се получава абсолютен адрес
- Полученият адрес се сравнява със стойността в крайния регистър
- Ако адресът е извън границите на крайния регистър, ОС генерира прекъсване

Разделяне на страници

- Разделете паметта на малки части с еднакъв размер и разделете всеки процес на същите по размер части

- Частите на процеса се наричат ***страници***
- Частите от паметта се наричат ***кадри***

Разделяне на страници

- Операционната система съхранява таблица със страниците за всеки процес
  - Съдържа местоположението на кадъра за всяка страница от процеса
  - Адресът в паметта се състои от номер на страница и отместване (офсет) вътре в страницата

Процеси и кадри

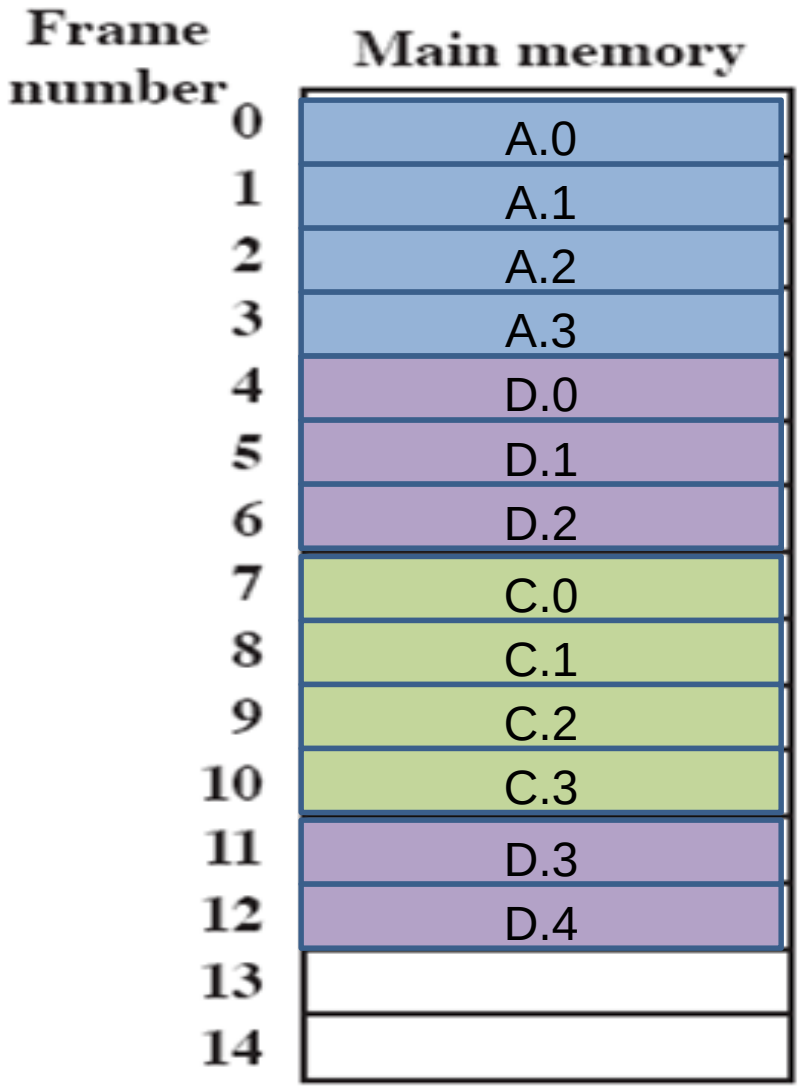


Таблица на страниците

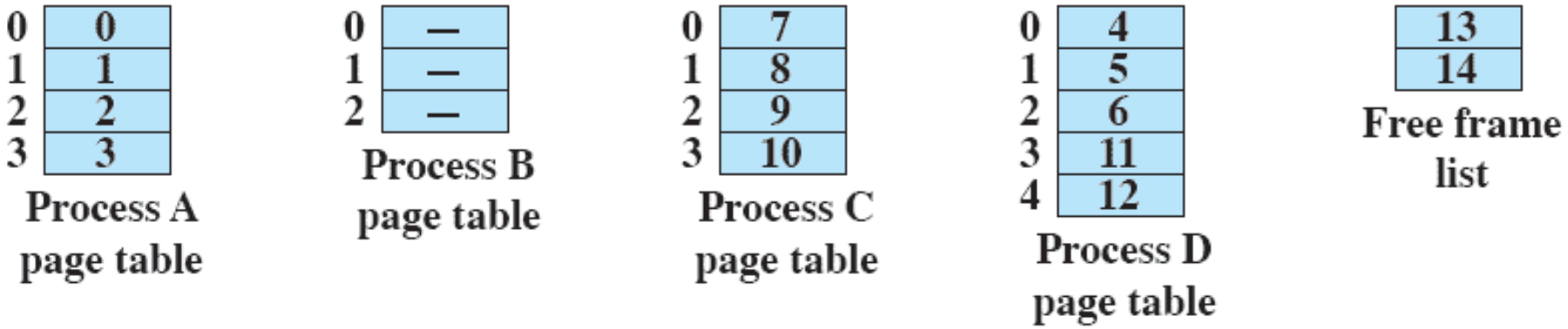


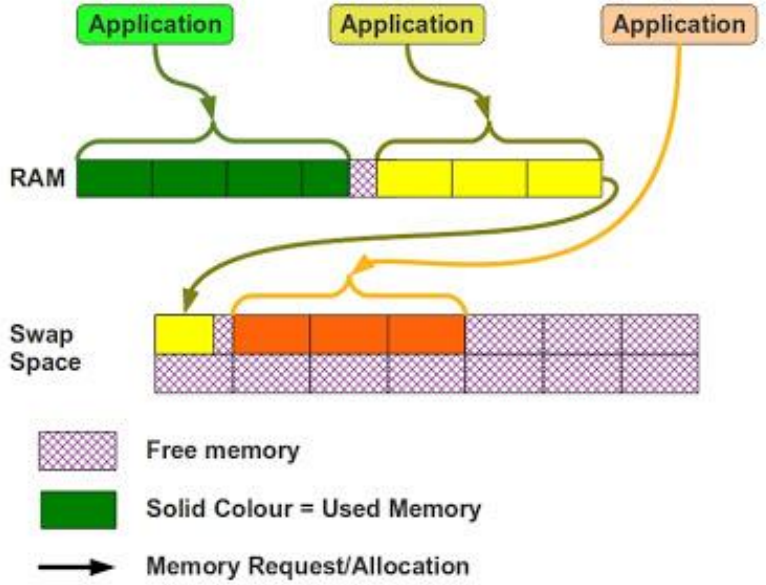
Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Сегментиране

- Една програма може да се подраздели на сегменти
  - Сегментите могат да са с различна дължина
  - Има максимална дължина на сегмента
- Адресирането има 2 части
  - Номер на сегмента и

  отместване
- Сегментирането е подобно на динамичното разделяне на дялове

Виртуална памет



- Виртуалната памет комбинира оперативната и използва временно свободно място на твърдия диск. Когато оперативната памет е малко, виртуалната памет премества данните от оперативната във файла за виртуална памет. Преместването на данните от и във файла за виртуална памет освобождава оперативна памет, което спомага за изпълнението на задачата от компютъра.
- Колкото повече оперативна памет има компютърът, толкова по-бързо работят програмите. Ако поради липса на оперативна памет, компютърът работи бавно, може да я компенсирате, като увеличите размера на виртуалната памет. Компютърът обаче чете данните от оперативната памет много по-бързо отколкото от твърдия диск, поради което добавянето на оперативна памет е по-добро решение.

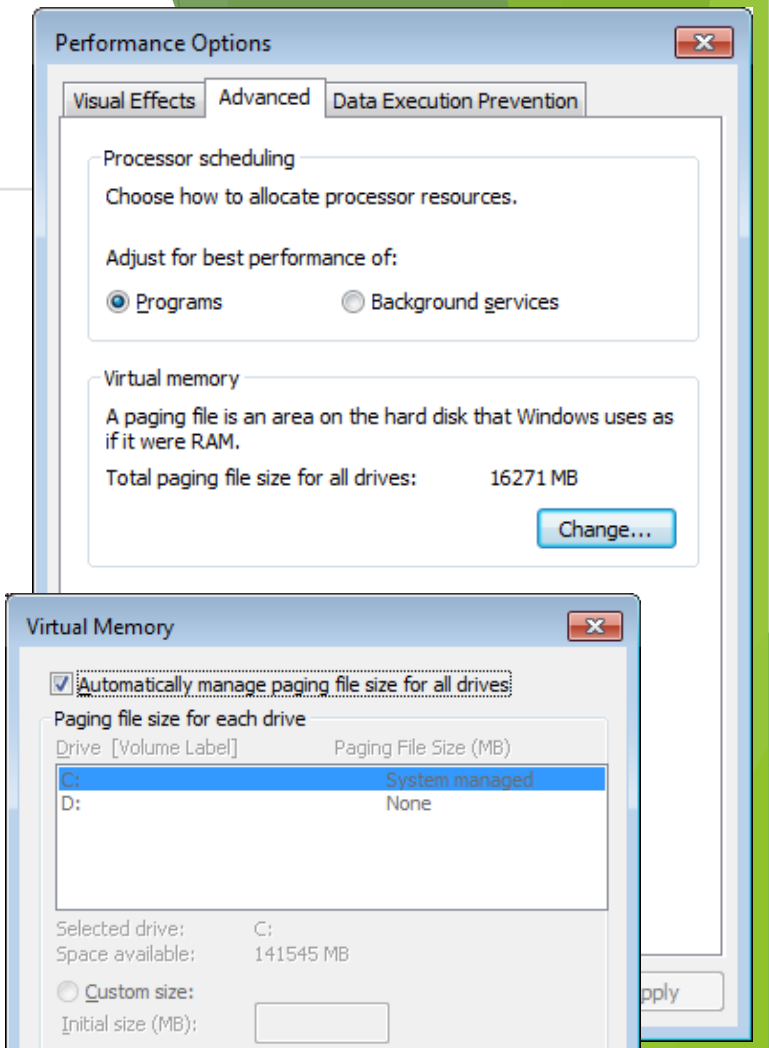
Виртуална памет и съобщения за грешка

Ако се покажат съобщения за малко виртуална памет, трябва или да добавите още оперативна, или да увеличите размера на файла за виртуална памет, за да могат да работят програмите на компютъра. Windows обикновено автоматично управлява размера, но той може да се зададе и ръчно, ако стойността по подразбиране не е достатъчна

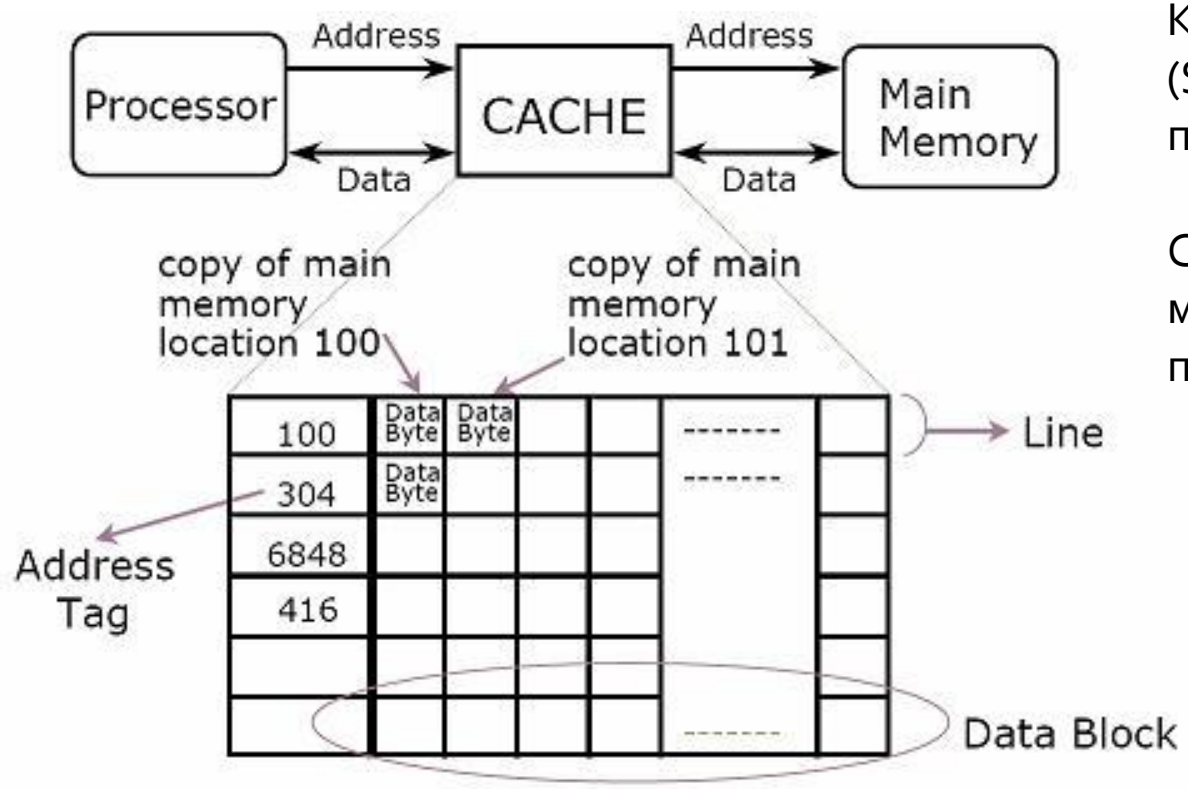
Промяна на размера на виртуалната памет

Ако получите предупреждение, че виртуалната памет е малко, трябва да увеличите минималния размер на файла за виртуална памет. Windows задава за първоначален минимален размер обема на оперативната памет (RAM), инсталирана на компютъра, с максимален размер 3 пъти над инсталираната оперативна памет. Ако предупреждения се показват и при препоръчаните нива, увеличите минималния и максималния размер.

1. Отворете "Система", като щракнете върху бутона **Старт**, щракнете с десния бутон върху **Компютър**, а след това изберете **Свойства**.
2. Отляво на екрана изберете **Разширени системни настройки**. Ако се покаже подкана да въведете администраторска парола или потвърждение, въведете паролата или потвърдете.
3. В раздела **Разширени** под **Производителност** щракнете върху **Настройки**.
4. Щракнете върху раздела **Разширени** и под **Виртуална памет** щракнете върху **Промени**.
5. Изчистете квадратчето **Автоматично определяне на размера на файла за виртуална памет за всички устройства**.
6. Под **Дисково устройство [етикет на том]** щракнете върху устройството, в което се намира файлът за виртуална памет, който искате да промените.
7. Щракнете върху **Размер по избор**, въведете новия размер в мегабайти в полето **Първоначален размер (МБ)** или **Максимален размер (МБ)** и щракнете върху **Задай**, след което върху **ОК**.



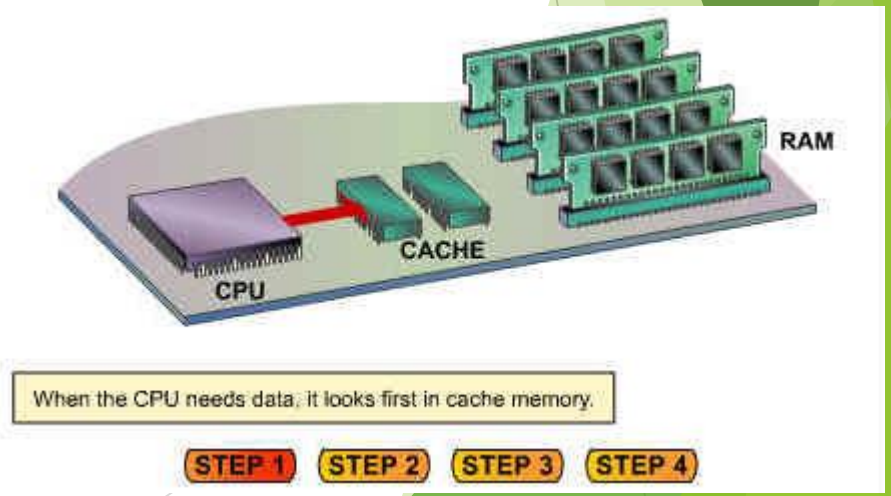
Кеш - ПАМЕТ



Кеш паметта е малка, високопроизводителна памет, която съхранява най-често използваните команди или данни.

Кеш паметта представа бързодействащо (SRAM) запомнящо устройство, което е поместено в процесора или извън него.

Служи като високоскоростен буфер между процесора и относително бавната оперативна памет.



РЕГИСТРИ

Регистри с общо предназначение.

Достъпни са за всички програми.

Контролни регистри.

Използват се от процесора за да контролират работата му.

Използват се от операционната система за да контролират изпълнението на приложните програми.

Регистри с общо предназначение

Достъпни са за всички програми.
Могат да се адресират като се използва асемблер.

Има два типа:

- Регистри за данни.
- Адресни регистри:
  - Използват се за реализация на различни схеми за адресация.
  - Индексен, сегментен, указател на стека

Контролни регистри

Не са достъпни директно.

Програмен брояч (Program Counter – PC).

Съдържа адреса на следващата инструкция, която трябва да бъде извлечена.

Регистър за инструкция (Instruction Register – IR).

Съдържа последната извлечена от паметта инструкция.

Регистър на състоянието (Program Status Word – PSW).

Резултат от сравнения.

Разрешаване и забрана на прекъсванията.

Потребителски режим и защитен режим на процесора.

Процесорни регистри

- По-бързи и по-малки от основната памет
- Видими регистри за потребителите
  - Дава възможност на програмиста да сведе до минимум реферирането до паметта чрез използването регистър
- Контрол и състояние на регистрите
  - Използва се от процесора за контрол на работа на самия процесора

Регистри – видими за потребителите

- Могат да бъдат реферирани с машинни езици
  - Достъпни от системни програми и приложни програми
- Съхраняват:
  - данни
  - адреси
  - регистри с условен код

Данни и адресни регистри

- Данни
  - С общо предназначение
  - Прилагат се някои ограничения
- Адресни
  - Индексни регистри
  - Сегментен указател
  - Указател на стек

Контрол и Статус регистри

- Програмен брояч (PC)
- Съдържа адресите на прочетениет инструкции
- Регистър на инструкции (IR)
  - Съдържа най-скоро прочетените
- Program status word (PSW)
  - Информация за статуса

История – MS OS

1981 – DOS 1.0 – Intel 8086

1983 – DOS 2.0

1984 – DOS 3.0 – Intel 80286

1980 – MS start GUI development

1990 – First GUI - Windows 3.0

1993 - Window NT 3.1

Windows 95,98, 2000

Windows Server 2003 – 64-bit

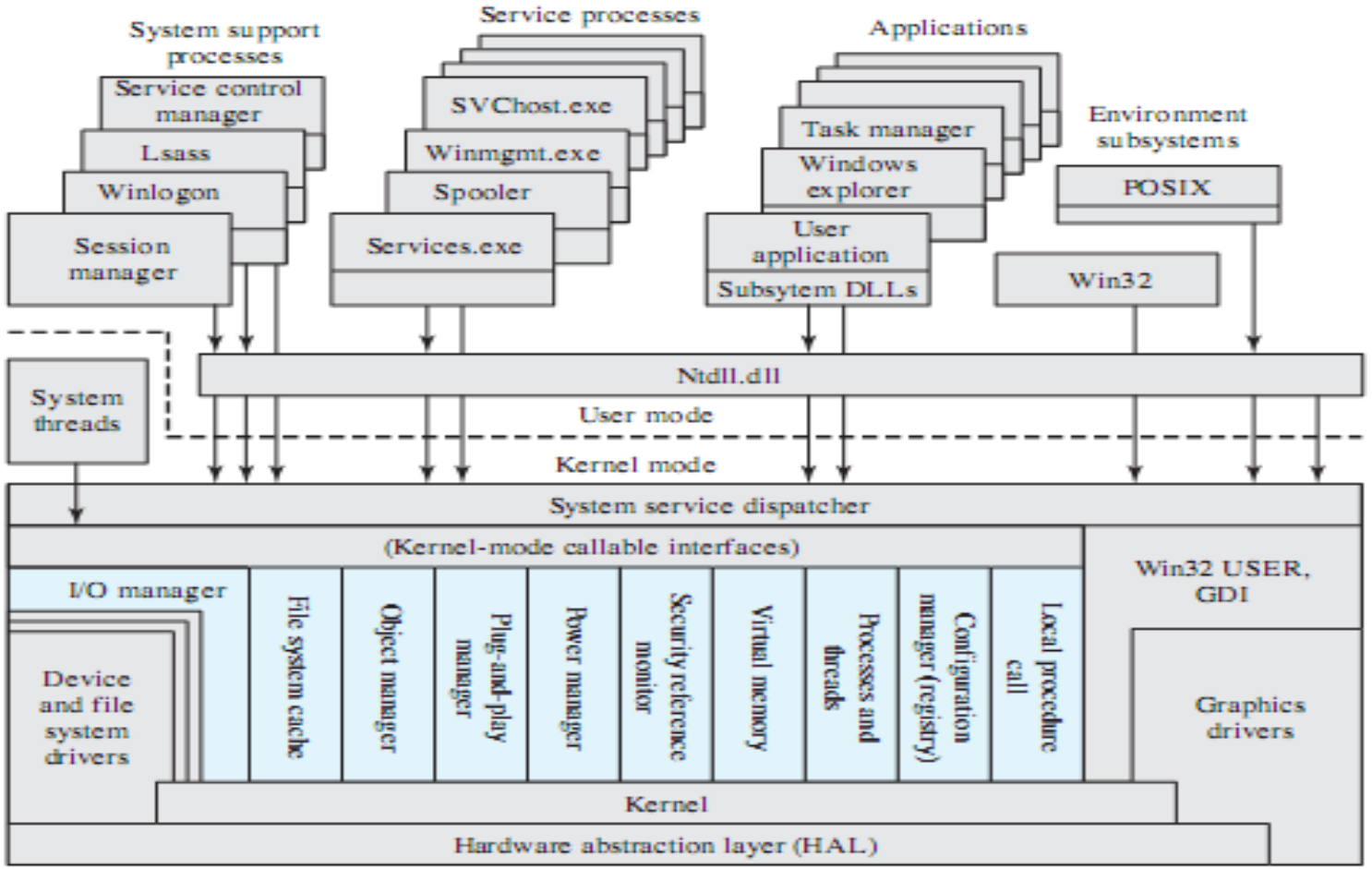
Windows XP, Vista, Server 2008, Windows 7, Windows 8, 8.1

2014 – Септември – Windows 10

Етапи при зареждане на Windows

- Проверка на хардуера
- Проверка на захранването
- Зареждане на BIOS
- BIOS POST(**Power-On Self-Test**)
- BIOS PnP (Plug and Play)
- BIOS boot device
- Зареждане на boot записите
- Master Boot record
- Partition Boot Record

Windows Architecture

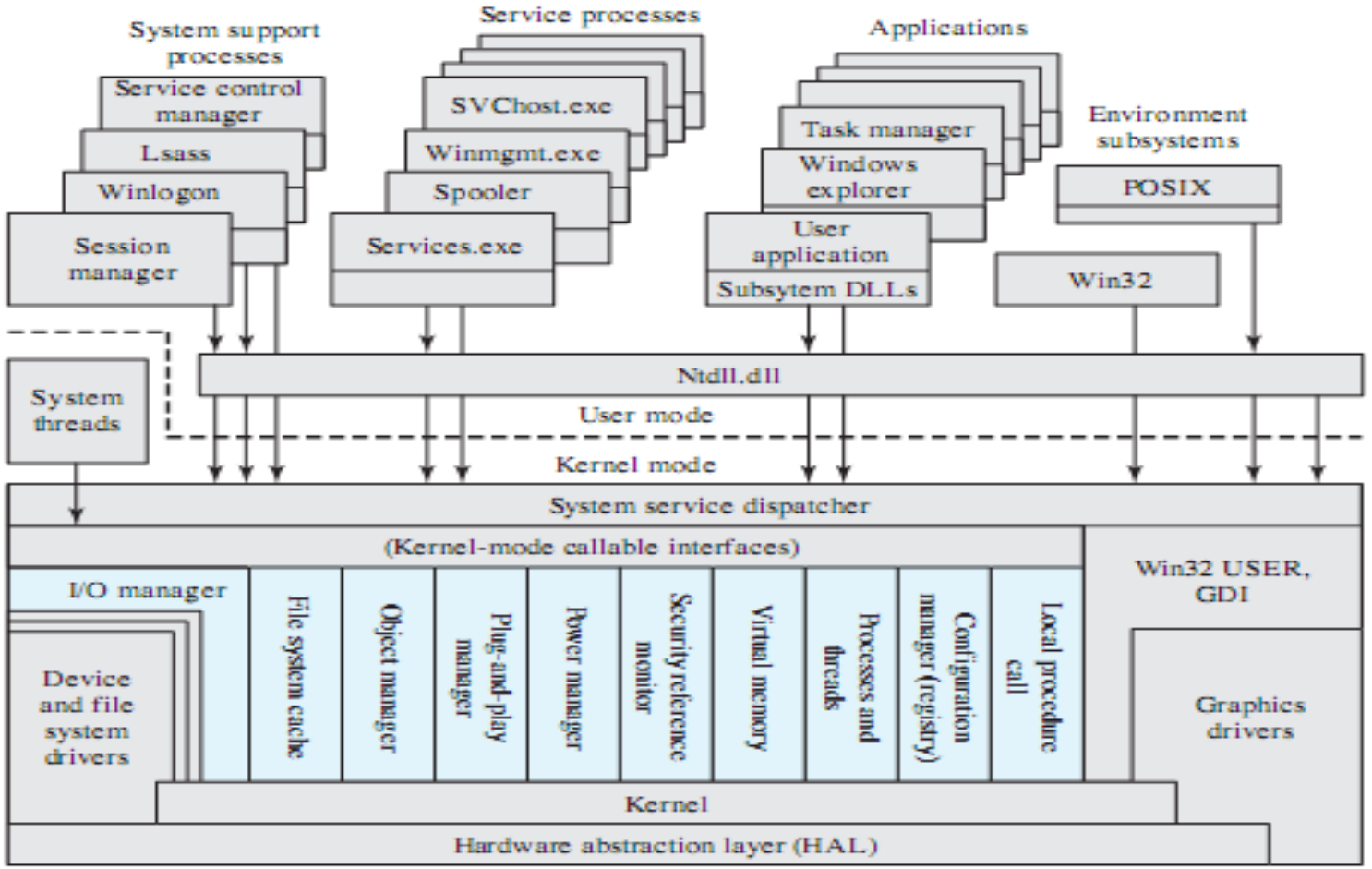


Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

Figure 2.13 Windows and Windows Vista Architecture [RUSS05]

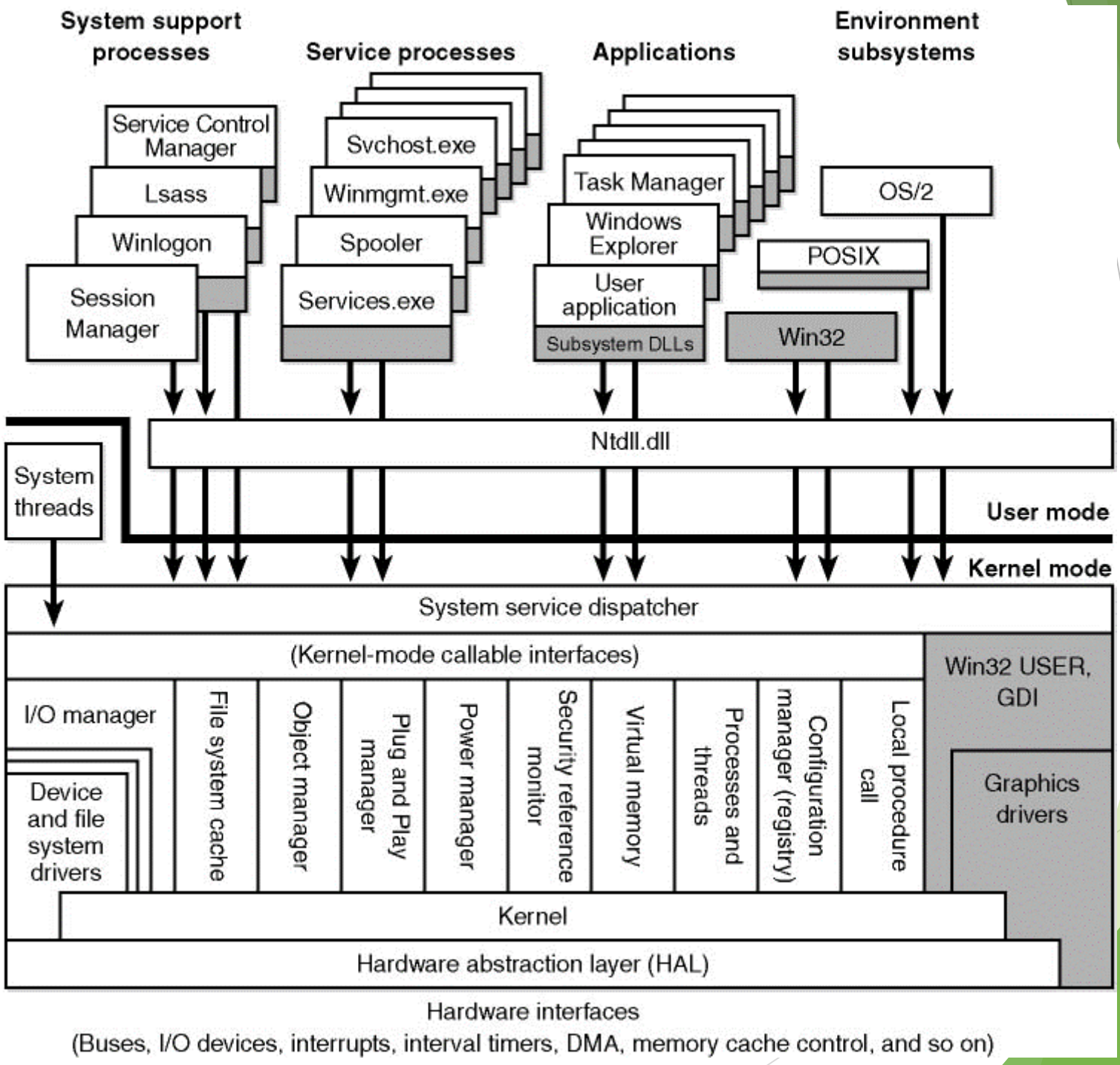
Windows Architecture



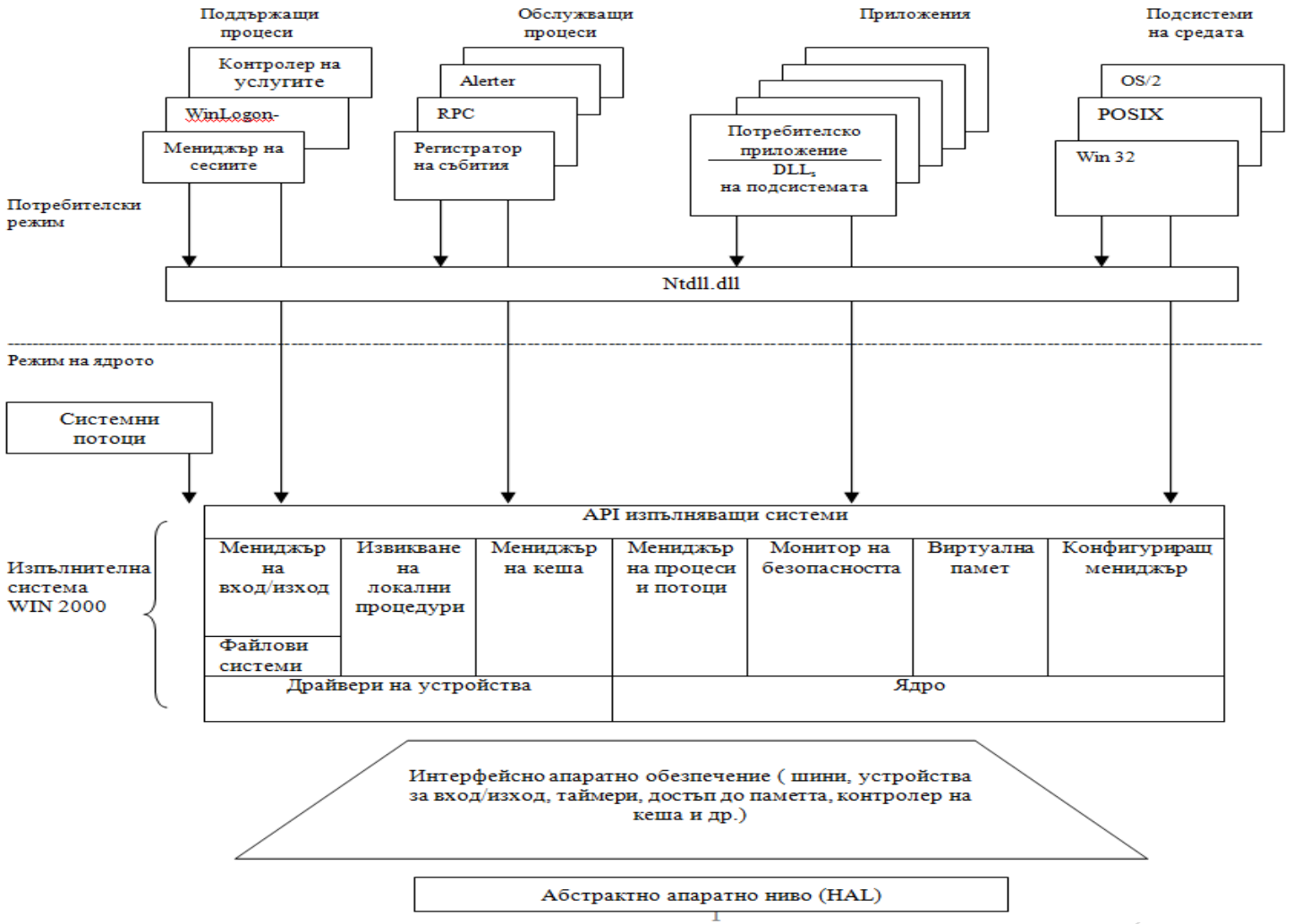
Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

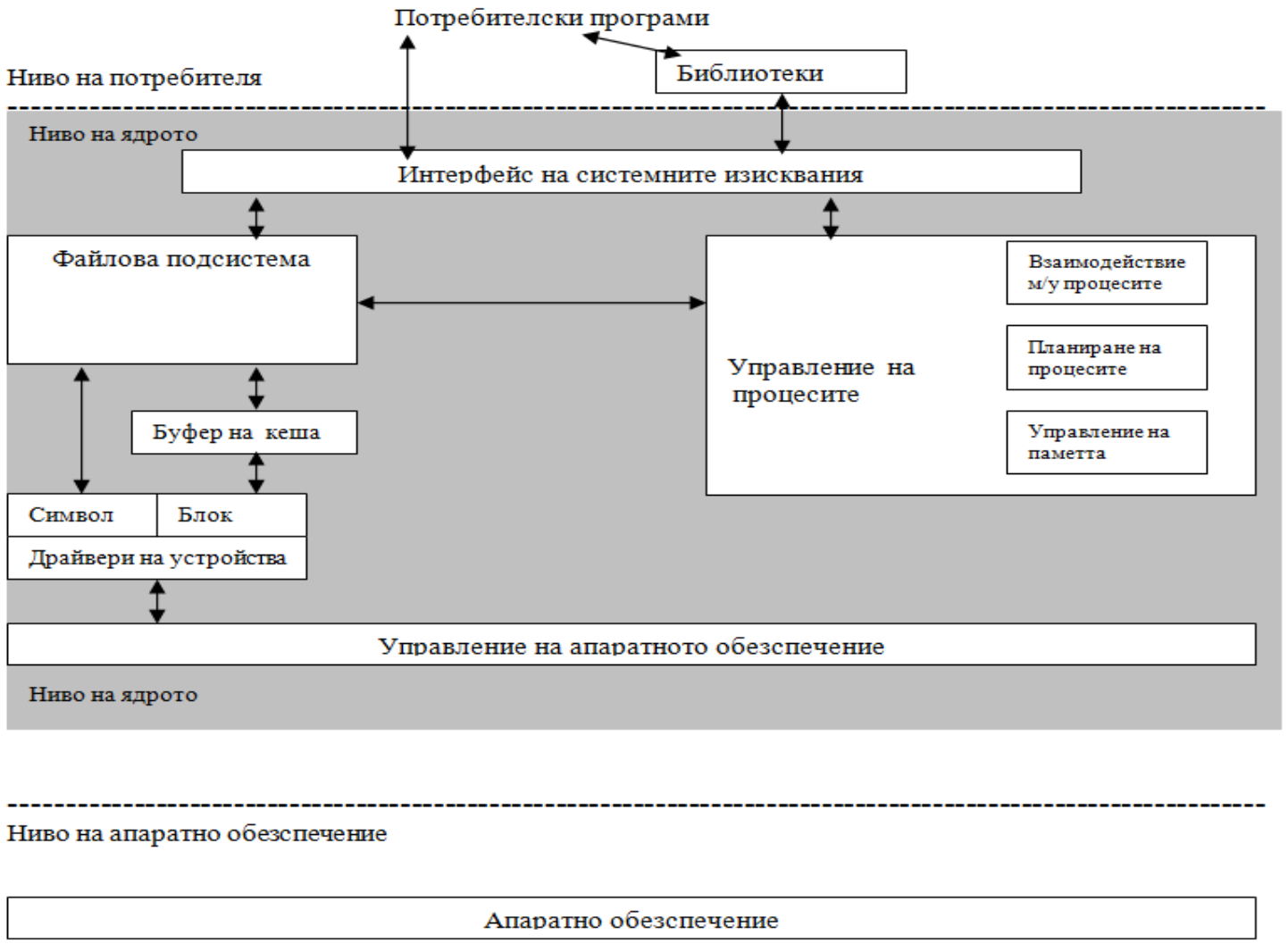
Figure 2.13 Windows and Windows Vista Architecture [RUSS05]


System support processes
Service processes
Applications
Environment subsystems
Service Control Manager
Lsass
Winlogon
Session Manager
Svchost.exe
Winmgmt.exe
Spooler
Services.exe
Task Manager
Windows Explorer
User application
Subsystem DLLs
OS/2
POSIX
Win32
Ntdll.dll
System threads
User mode
Kernel mode
System service dispatcher
(Kernel-mode callable interfaces)
I/O manager
Device and file system drivers
File system cache
Object manager
Plug and Play manager
Power manager
Security reference monitor
Virtual memory
Processes and threads
Configuration manager (registry)
Local procedure call
Win32 USER, GDI
Graphics drivers
Kernel
Hardware abstraction layer (HAL)
Hardware interfaces
(Buses, I/O devices, interrupts, interval timers, DMA, memory cache control, and so on)

Windows ядро



Ядро Unix


Потребителски програми
Библиотеки
Ниво на потребителя
Ниво на ядрото
Интерфейс на системните изисквания
Файлова подсистема
Взаимодействие м/у процесите
Планиране на процесите
Управление на процесите
Управление на паметта
Буфер на кеша
Символ
Блок
Драйвери на устройства
Управление на апаратното обезспечение
Ниво на ядрото
Ниво на апаратно обезспечение
Апаратно обезспечение

История - UNIX

1970 – at Bell Labs (C)

1976 – Version 6

1978 – Version 7

1982 – UNIX System III

(Berkeley Software Distribution) BSD

FreeBSD – Internet Based servers

FreeBSD 5.0– MAC OS X

Solaris 10

Description of UNIX

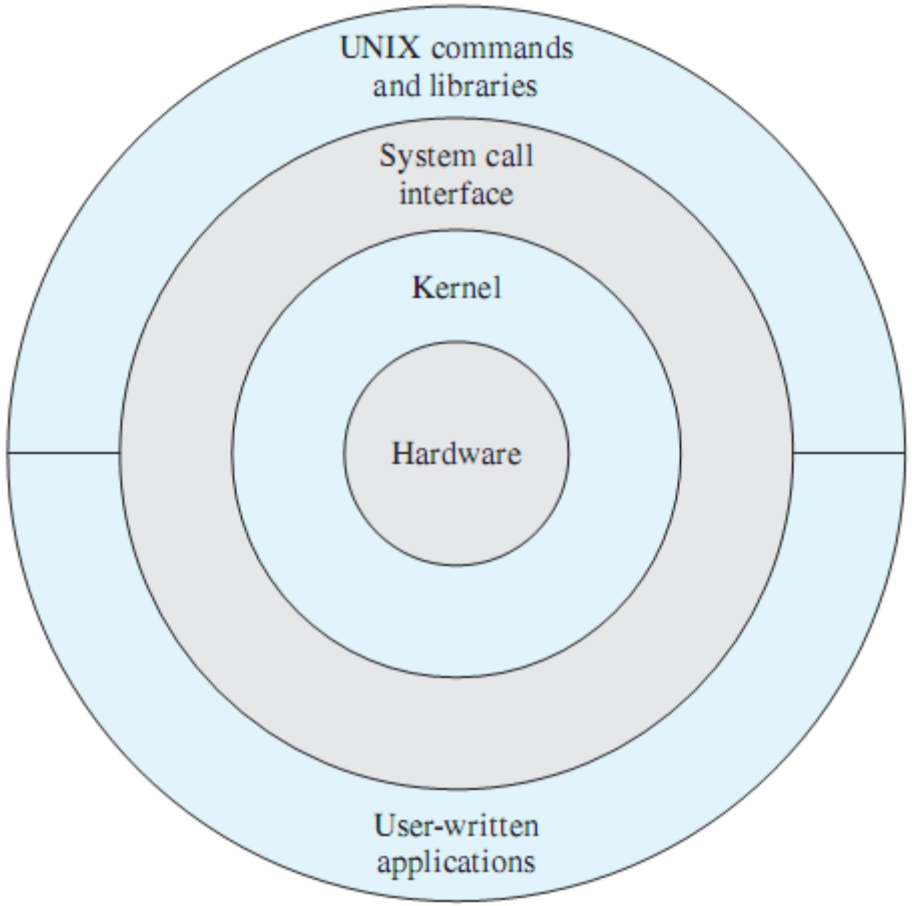


Figure 2.14 **General UNIX Architecture**

Traditional UNIX Kernel

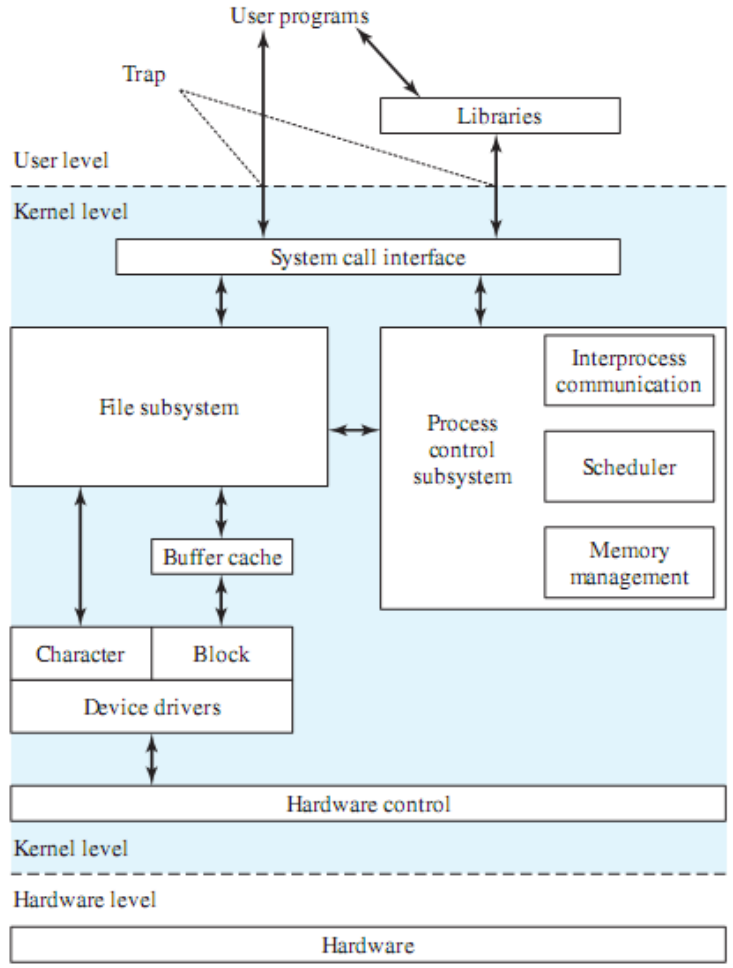


Figure 2.15 **Traditional UNIX Kernel**

Linux - История

- 1991 – Linux – Linus Torvals – Unix based

Intel 80386

- Linux 2.6
- Linux 3.16.3
- Linux 4.2

Разработката по първата версия на ядрото на Linux започва през 1991, когато студентът Линус Торвалдс решава да създаде своя операционна система, базирана на UNIX. Създадената операционна система е пусната с отворен код под лиценза на GNU.

Структура на ядрото

Линукс ядрото е представител на монолитните ядра. Монолитно ядро е ядро, което в себе си обединява разнообразни видове функции като:

- Драйвери на устройствата
- Управление на паметта, управление на разписанието на процесите
- Между процесна комуникация
- Поддръжка на файловите системи
- Поддръжка на виртуални файлови системи
- Системни изисквания

ДРАЙВЕРИ НА УСТРОЙСТВАТА

В ядрото на Линукс се поддържат голям брой устройства – от поддръжката на основни за компютъра като процесор, основни интерфейси на процесора за свързване с други устройства на системата, управление на директния достъп до паметта (DMA) и други; до различни периферни устройства – оптични устройства, система за управление на захранването, мрежови адаптери и т.н.

Драйверите на периферните устройства могат да бъдат вградени в ядрото, както и могат да бъдат зареждани модулни (при нужда). Това позволява гъвкаво системата да бъде настроена за работа по различни задачи.

Управление на паметта

Управлението на паметта е една от основните дейности на операционната система. Тази дейност е вградена в ядрото на Линукс. Управлението на паметта представлява процес по разпределението на паметта като ресурс между различните процеси като създава виртуално адресно пространство за всеки един процес. По този начин всеки един процес използва паметта, но не достига директно със своите адреси до физическите клетки на паметта, а се обръща към операционната система, която ги преобразува до реално. Ядрото се грижи да заделя, освобождава и дефрагментира паметта, като това са услуги, които се използват при извикване на функции като malloc() и free() за програмни езици като C, и new и delete за езици като C++.

Управлението на разписанието на процесите

- Това е още една от основните функции на операционните системи, която е вградена в ядрото. Управлението на разписанието на процесите е характерна за многозадачните ОС. Тази подсистема решава кой процес кога да бъде изпълняван.
- В съвременните компютърни системи е рядкост броя на процесите да бъдат колкото броят на процесорите (и ядрата), за това е необходимо да се въведе подсистема, която да управлява опашката от задачи за изпълнението им от процесора. Алгоритмите на редуване на задачите са разнообразни, дори и за една операционна система, като много често се избира в зависимост от целите, за които тази система ще бъде използвана.

МЕЖДУПРОЦЕСНА КОМУНИКАЦИЯ (IPC)

- Това е възможност на процесите да обменят директно информация. IPC се използва в много от съвременните сложни програмни архитектури, където върху една задача работят множество процеси. Това им налага нуждата да общуват помежду си като обменят текущи решения.
- Междупроцесната комуникация е необходима, защото процесите нямат споделено адресно пространство и следователно не могат да записват в един и същи блок памет. Вместо това те общуват посредством сигнали и пайпове (pipes).
- Сигналите са форма на междупроцесна комуникация, при която процесите обменят кратки съобщения (всяко от които представлява код). Сигналите се обработват асинхронно и могат да послужат за външно извеждане на процеса от дадено състояние.
- Pipes се използват, когато е необходимо предаването на по-големи структури от данни, низове и други. За тази цел между два процеса може да бъде отворен канал за обмен на данните (pipe). Всеки процес може да записва данни в този канал, а от другата страна те могат да бъдат прочетени, когато е възможно от другия процес, без да ми се спира работата.

Поддръжка на файлови системи

- Файловата система е неразделна част от операционната система, затова тя е част от ядрото ѝ. Файловата система представлява система от правила и структури за управлението на данните, записани на носителите на компютърната система (енергозависими и енергонезависими памети).
- Създването, четенето, записът, управлението на достъпа на файл са услуги на операционната система. Тя предоставя на потребителите достъп до данните им без те да се интересуват как точно са представени в носителя.

Виртуални файлови системи

Достъпът до устройствата в Линукс операционната система се осъществява посредством файлове, представени в структурата на файловата система. По този начин работата с периферни устройства се свежда до работа с файлове.

Системни извиквания

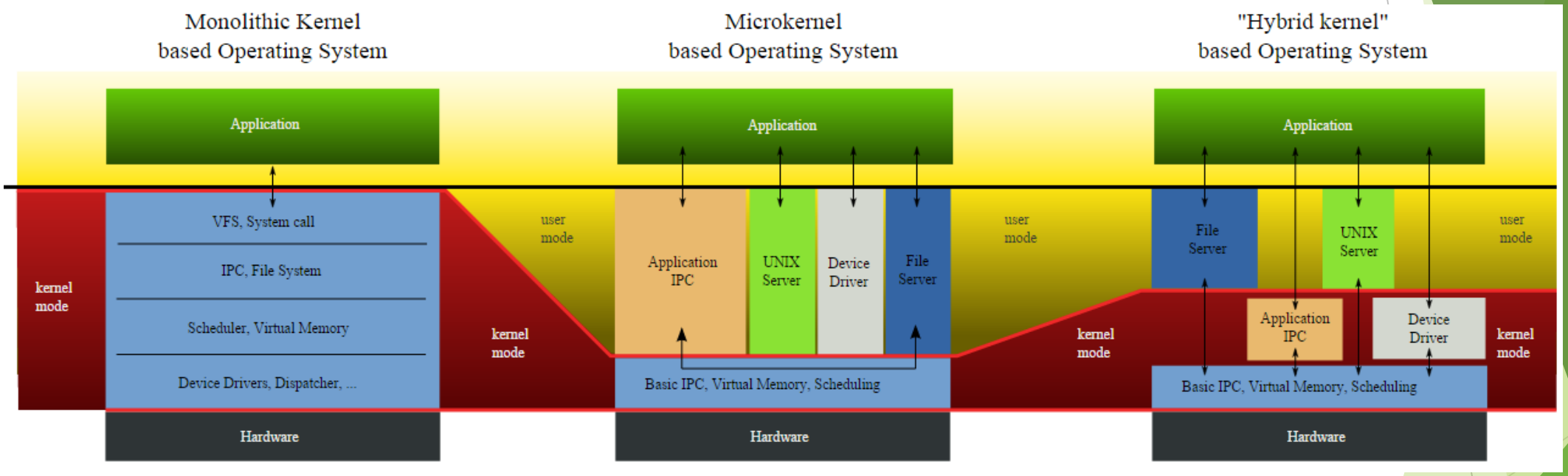
За комуникацията на процесите с ядрото се използват системни извиквания, те са представени в преносимия интерфейс на операционната система (POSIX – Portable Operating System Interface).

В него са дефинирани множество системни примитиви за работа с ядрото като функции за работа с файлове, създаване на канали за обмен на информация, за управление на паметта, създаване и поддръжка на мрежови връзки и др.

ЗАКЛЮЧЕНИЕ

Ядрото на операционната система включва всички основни дейности на операционната система, които ѝ позволяват да надстрои хардуерното ниво като въведе правила за управлението на ресурсите и предостави на потребителските програми да използват ресурси без да се създава конфликт с другите процеси. Ядрото на Линукс е гъвкаво и може да се вгради в системи с различни видове архитектура, защото е продукт с отворен код и може да се допълва, обновява и настройва непрекъснато.

Linux Kernel



Linux Kernel Modules

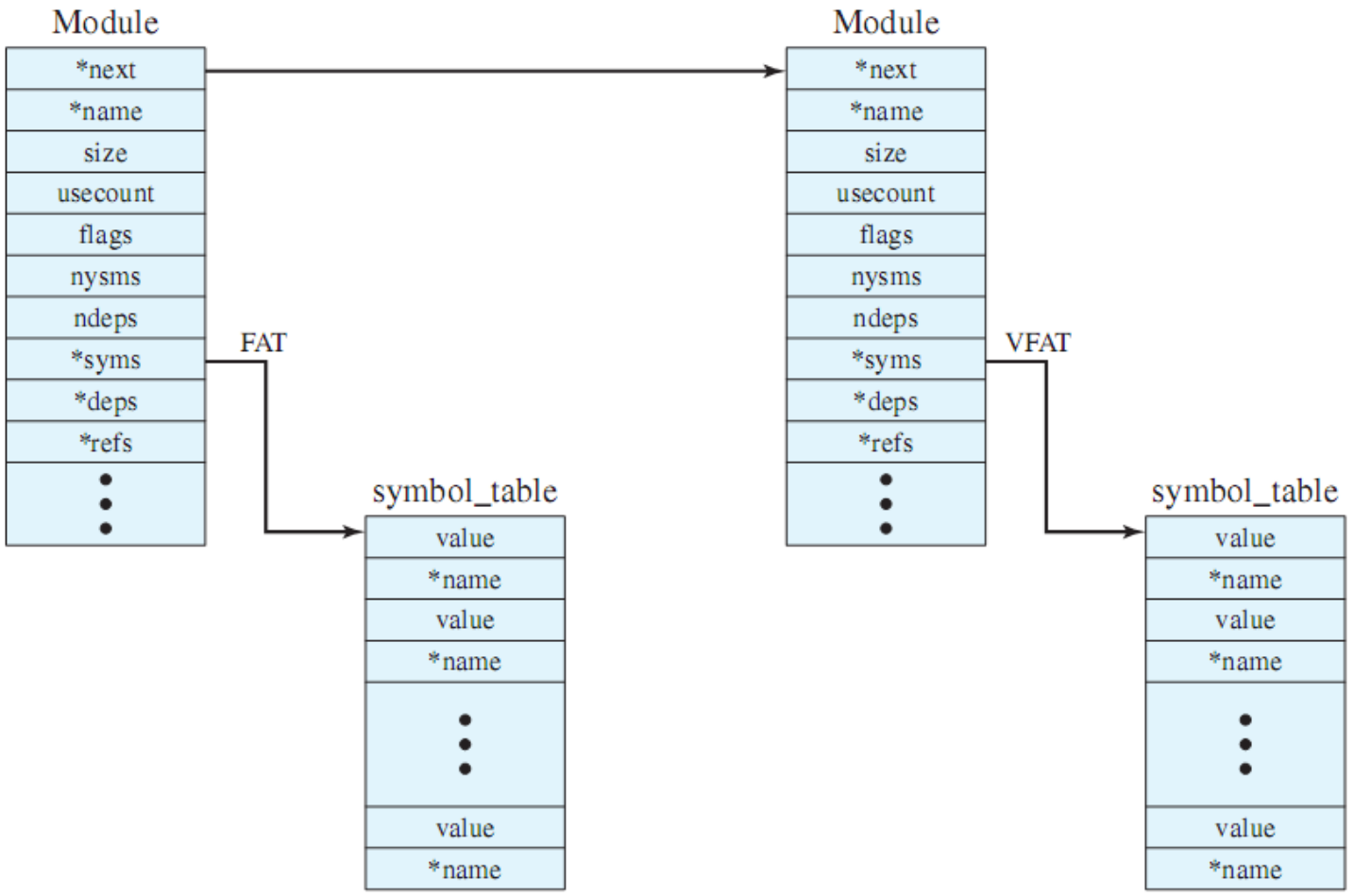


Figure 2.17 **Example List of Linux Kernel Modules**

Linux Kernel Components

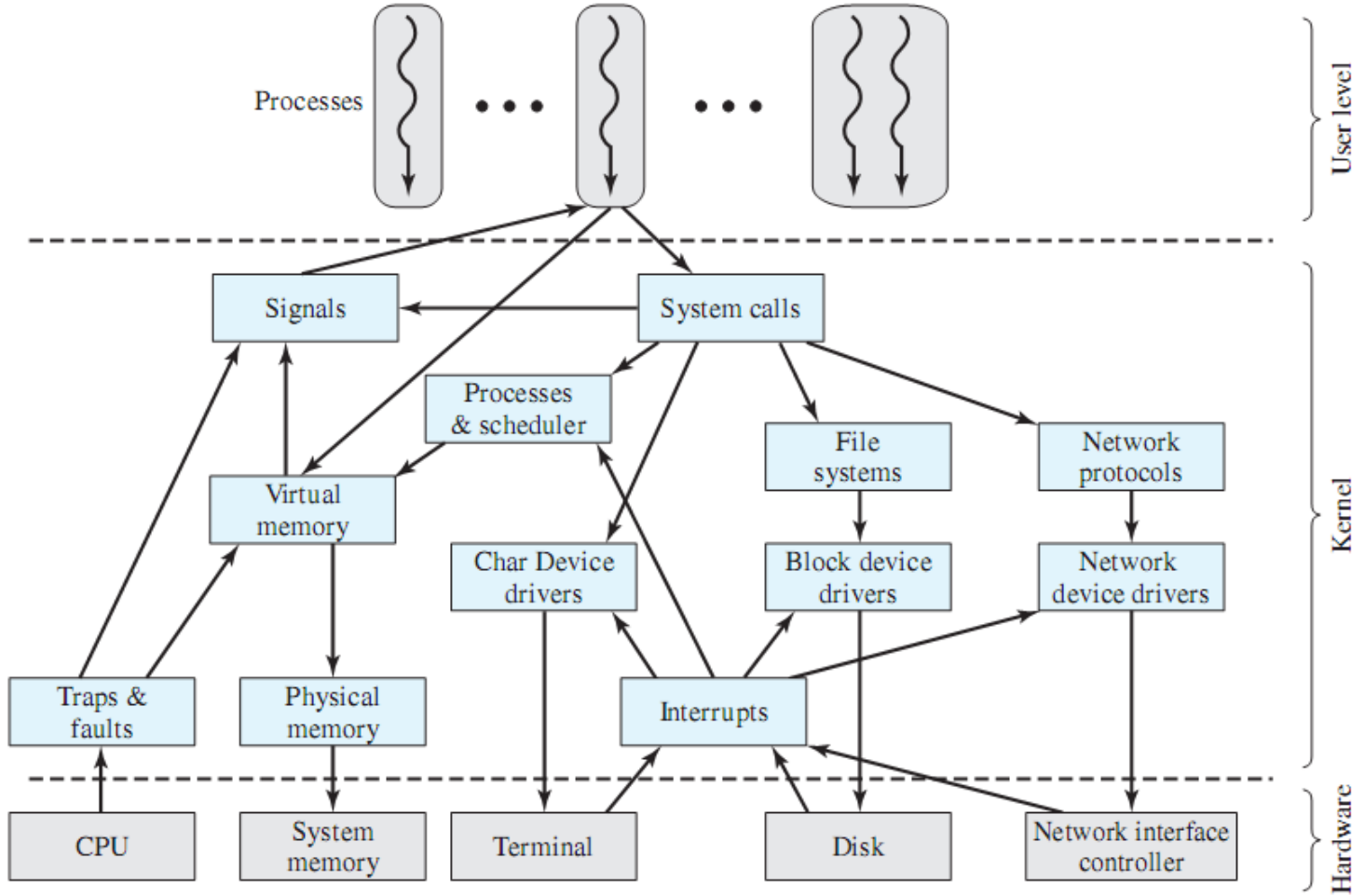
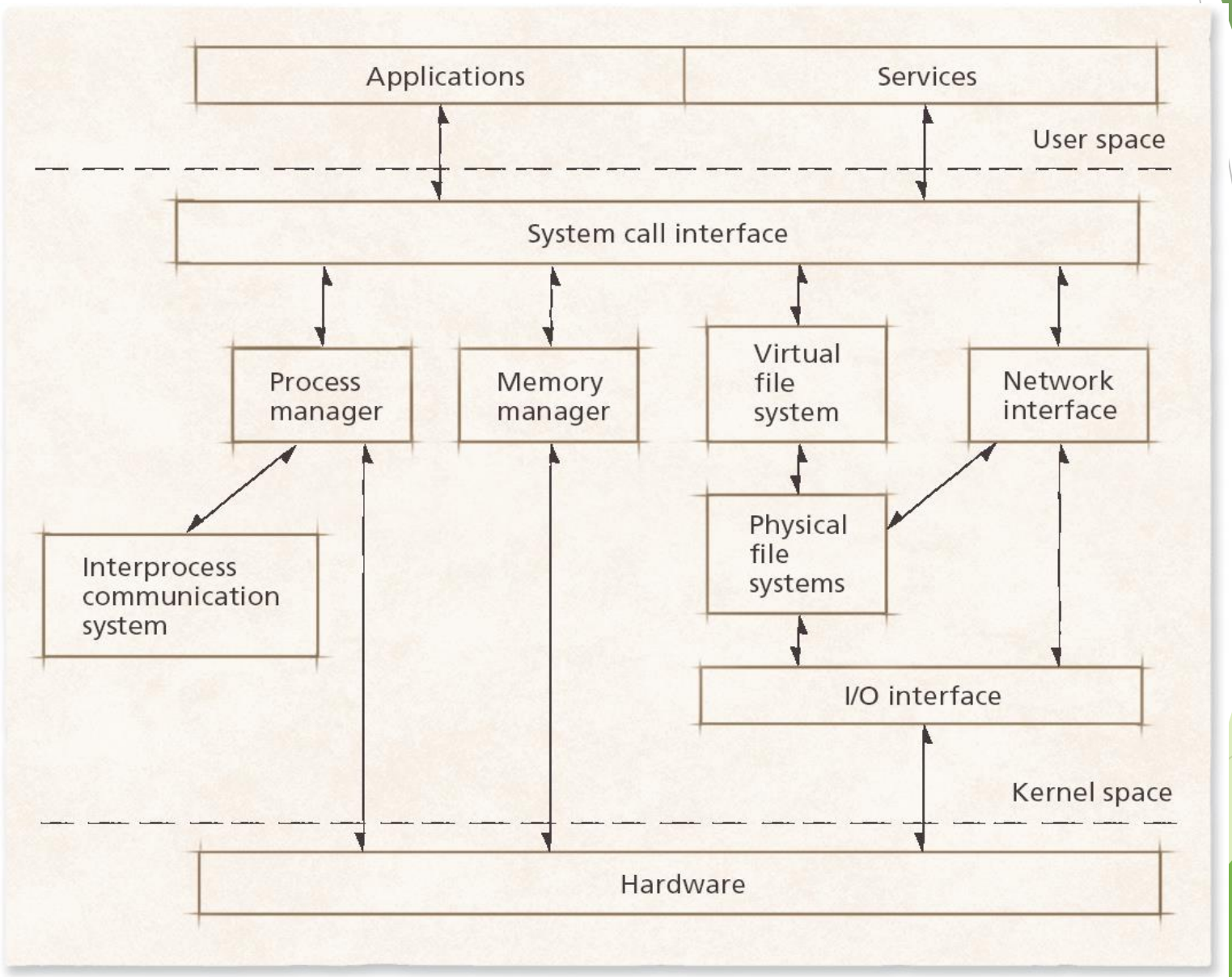
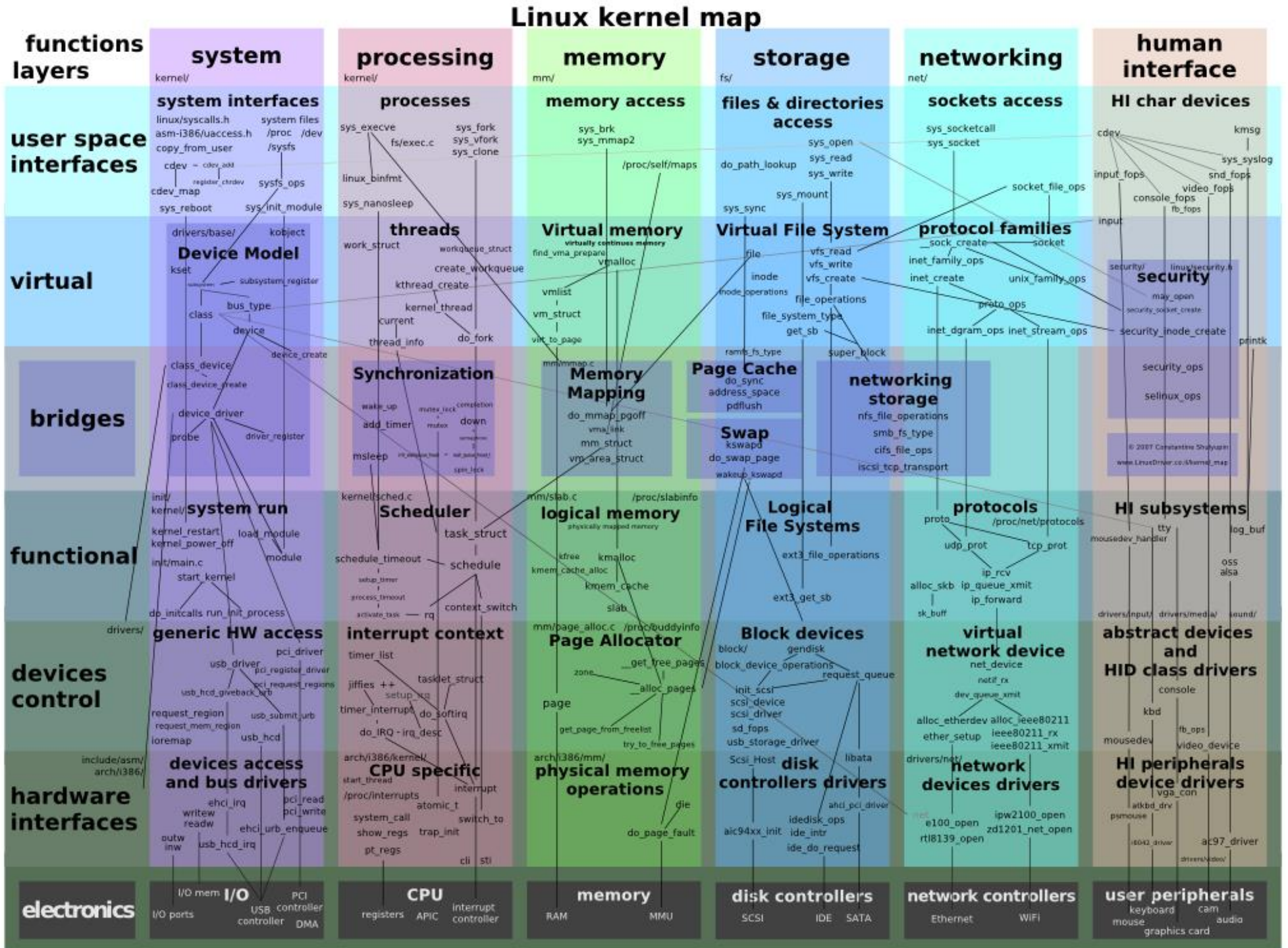


Figure 2.18 **Linux Kernel Components**


Applications
Services
User space
System call interface
Process manager
Memory manager
Virtual file system
Network interface
Interprocess communication system
Physical file systems
I/O interface
Kernel space
Hardware


Linux kernel map
functions
layers
system
kernel/
processing
kernel/
memory
mm/
storage
fs/
networking
net/
human interface
user space interfaces
system interfaces
linux/syscalls.h
asm-i386/uaccess.h
copy_from_user
system files
/proc
/dev
/sysfs
cdev
cdev_add
register_chrdev
cdev_map
sysfs_ops
sys_reboot
sys_init_module
processes
sys_execve
fs/exec.c
sys_fork
sys_vfork
sys_clone
linux_binfmt
sys_nanosleep
memory access
sys_brk
sys_mmap2
/proc/self/maps
files & directories access
sys_open
sys_read
sys_write
do_path_lookup
sys_mount
sys_sync
sockets access
sys_socketcall
sys_socket
socket_file_ops
HI char devices
cdev
kmsg
sys_syslog
input_fops
snd_fops
video_fops
console_fops
fb_fops
input
virtual
drivers/base/
kobject
Device Model
kset
subsystem_register
bus_type
class
device
device_create
threads
work_struct
workqueue_struct
create_workqueue
kthread_create
kernel_thread
current
thread_info
do_fork
Virtual memory
virtually continuous memory
find_vma_prepare
vmalloc
vmlist
vm_struct
virt_to_page
mm/mmap.c
Virtual File System
file
vfs_read
vfs_write
vfs_create
inode
inode_operations
file_operations
file_system_type
get_sb
ramfs_fs_type
super_block
protocol families
__sock_create
socket
inet_family_ops
inet_create
unix_family_ops
proto_ops
inet_dgram_ops
inet_stream_ops
security/
linux/security.h
security
may_open
security_socket_create
security_inode_create
security_ops
selinux_ops
printk
bridges
class_device
class_device_create
device_driver
probe
driver_register
Synchronization
wake_up
mutex_lock
completion
add_timer
mutex
down
msleep
Memory Mapping
do_mmap_pgoff
vma_link
mm_struct
vm_area_struct
Page Cache
do_sync
address_space
pdflush
Swap
kswapd
do_swap_page
wakeup_kswapd
networking storage
nfs_file_operations
smb_fs_type
cifs_file_ops
iscsi_tcp_transport
© 2007 Constantine Shulyupin
www.LinuxDriver.co.il/kernel_map
functional
init/
kernel/
system run
kernel_restart
kernel_power_off
load_module
module
init/main.c
start_kernel
do_initcalls
run_init_process
kernel/sched.c
Scheduler
task_struct
schedule_timeout
schedule
setup_timer
process_timeout
activate_task
rq
context_switch
mm/slab.c
/proc/slabinfo
logical memory
physically mapped memory
kfree
kmalloc
kmem_cache_alloc
kmem_cache
slab
Logical File Systems
ext3_file_operations
ext3_get_sb
protocols
proto
/proc/net/protocols
udp_prot
tcp_prot
ip_rcv
alloc_skb
ip_queue_xmit
ip_forward
sk_buff
HI subsystems
tty
log_buf
mousedev_handler
oss
alsa
drivers/input/
drivers/media/
sound/
devices control
drivers/
generic HW access
pci_driver
usb_driver
pci_register_driver
pci_request_regions
usb_hcd_giveback_urb
request_region
request_mem_region
ioremap
usb_submit_urb
usb_hcd
interrupt context
timer_list
jiffies ++
setup_irq
tasklet_struct
timer_interrupt
do_softirq
do_IRQ - irq_desc
mm/page_alloc.c
/proc/buddyinfo
Page Allocator
__get_free_pages
zone
__alloc_pages
page
get_page_from_freelist
try_to_free_pages
Block devices
block/
gendisk
block_device_operations
request_queue
init_scsi
scsi_device
scsi_driver
sd_fops
usb_storage_driver
virtual network device
net_device
netif_rx
dev_queue_xmit
alloc_etherdev
alloc_ieee80211
ether_setup
ieee80211_rx
ieee80211_xmit
abstract devices and HID class drivers
console
kbd
mousedev
fb_ops
video_device
hardware interfaces
include/asm/
arch/i386/
devices access and bus drivers
ehci_irq
writew
readw
pci_read
pci_write
ehci_urb_enqueue
outw
inw
usb_hcd_irq
arch/i386/kernel/
CPU specific
start_thread
/proc/interrupts
interrupt
atomic_t
system_call
switch_to
show_regs
trap_init
pt_regs
cli
sti
arch/i386/mm/
physical memory operations
die
do_page_fault
Scsi_Host
disk controllers drivers
libata
ahci_pci_driver
idedisk_ops
ide_intr
ide_do_request
aic94xx_init
drivers/net/
network devices drivers
ipw2100_open
zd1201_net_open
e100_open
rtl8139_open
HI peripherals device drivers
vga_con
atkbd_drv
psmouse
i8042_driver
ac97_driver
drivers/video/
electronics
I/O
I/O mem
I/O ports
USB controller
PCI controller
DMA
CPU
registers
APIC
interrupt controller
memory
RAM
MMU
disk controllers
SCSI
IDE
SATA
network controllers
Ethernet
WiFi
user peripherals
keyboard
cam
mouse
audio
graphics card

Inode

- Всеки файл се представя чрез структура, наречена индексен дескриптор (inode). Всеки дескриптор съдържа описание на файл:вид,права за достъп,автор,време на създаване, модифициране и достъп, указатели към блокове с данни. Когато се създава дисков дял в Gnu/Linux, системата създава фиксиран брой дескриптори, като този брой е и максималния брой файлове от всякакви видове които могат да съществуват по едно и също време на този дял от диска. Дескрипторът се изгражда при създаването на файла и се актуализира при работа с файла.

Директории

/ - Коренна директория

/bin - съдържа основните програми на операционната система, без които тя не би функционирала правилно. Повечето програми от тази директория може да се изпълняват от всички потребители. Тя присъства и в пътя по подразбиране на всички потребители.

/boot - съдържа ядрото на операционната система, конфигуцаионния файл на ядрото както и други файлове, необходими за коректно извършване дейността на ядрото.

/dev - съдържа файловете отговарящи за хардуерните устройства на компютъра. Файловете в тази директория са достъпни само за четене от потребителите. Пример е флопи дисковото устройство представено като fd0 или по-конкретно /dev/fd0

/mnt или /media - незадължителна системна директория. Тя се създава автоматично и нейната цел е обединяването на всички монтирани устройства на едно място. Линукс обаче не ви ограничава да монтирате устройства в други директории.

/opt - тук повечето дистрибуции инсталират графичните среди като KDE и GNOME.

/proc - това е директория с наличие на псевдофайлове. Тези псевдофайлове се използват от Gnu/Linux за извеждане на информация относно хардуерни настройки и параметри. Добър пример е псевдофайла /proc/meminfo който съдържа информация за наличната памета.

/root - домашната директория на суперпотребителя(root). За обикновения потребител тази директория е заключена и той няма никакъв достъп до файловете в нея.

/etc - съдържа глобалните конфигурационни файлове на операционната система и нейните програми. Някои от програмите може да пазят конфигурационни файлове и на други места като например: /usr/local/etc/

/home - съдържа home директории на потребителите. Тук за всеки потребител се създава отделна директория в която може да се съхраняват конфигурационни файлове или файлове от различен тип.

/lib - директорията съдържа най-важните споделени библиотеки на операционната система – glibc-solibs. Тези библиотеки се наричат споделени (shared object), защото повечето програми ги използват при изпълнението си.

/lib/modules - също много важна директория. Тук се съхраняват модулите на ядрото, т.е. драйверите на операционната система. Туй като всяка Линукс система може да има повече от едно ядро, то в тази директория се създават поддиректории с номера на всяко едно инсталирано ядро.

/mnt или /media - незадължителна системна директория. Тя се създава автоматично и нейната цел е обединяването на всички монтирани устройства на едно място. Линукс обаче не ви ограничава да монтирате устройства в други директории.

/tmp - директория в която програмите записват временни (temp) файлове.Тя е достъпна за абсолютно всеки потребител в системата.

/usr – Тук има голямо количество директории от инсталираните софтуерни приложения на системата. Приложенията са споделени т.е. Видими за всички потребители в системата.

/usr/bin - съдържа изпълнимите файлове на инсталираните приложения, които не са важни за коректното функциониране на системата.

/usr/doc - част от документацията на програмите. Тук се съхранява Linux-HOWTO.

/usr/include - тази директория е необходима само ако ще се инсталират програми от изходен код. Тук се пазят заглавните C файлове на инсталираните библиотеки и програми, които са необходими за инсталиране на други програми.

/usr/local - директория в която по подразбиране се инсталират програмите компилирани от изходен код.

/usr/man - най-мащабната част от Линукс документацията. Тук се съхранява ръководство (manual page) за почти всички инсталирани програми. Достъпа до тази информация се осъществява с програмата man.

/usr/lib - важна директория, в която се съхраняват повечето инсталирани библиотеки.

/usr/share - също важна директория. Тук програмите инсталират файловете, необходими за тяхната правилна работа.

/usr/src - в тази директория се пази изходният код на Линукс ядрото. Той е необходим само когато ядрото ще се прекомпилира. И тук подобно на /lib/modules може да има код на повече от едно ядро. Те се съхраняват в директории, различаващи се с номера на версията. В директорията /usr/src присъства и линк linux, който сочи към една от директориите съдържащи изходен код.

/var - Административна директория съдържаща предимно системни дневници(логове) на програмите и система, както и опашка с отложените или предстоящи задачи за автоматично изпълнение. Директория съдържа и друга информация, като пристигнали писма, файлове заключващи дадени процеси и сокети на работещи в момента програми.