# 15-213
*Introduction to Computer Systems*

# Memory Management II: Dynamic Storage Allocation Oct 13, 1998

## Topics
- other placement policies (cont)
- buddy systems
- implementation issues (sequential fits)



# Basic allocator mechanisms

## Sequential fits
- best fit, first fit, or next fit placement
- various splitting and coalescing options
  - splitting thresholds
  - immediate or deferred coalescing

## Segregated free lists
- simple segregated storage
- segregated fits
  - buddy systems



# Simple segregated storage

## Separate heap for different sized blocks
## No splitting
## To allocate a block of size n:
- if free list for size n is not empty,
  - allocate first block on list
- if free list is empty,
  - get a new page
  - create new free list
  - allocate first block on list
- constant time

## To free a block:
- Usual constant-time coalescing

## Tradeoffs:
- fast, but can fragment badly



# Segregated fits

## Array of free lists, each one for some size class
## To allocate a block of size n:
- search appropriate free list for block of size m > n
- if an appropriate block is found:
  - split block and place fragment on appropriate list (optional)
- if no block is found, try next larger class
- repeat until block is found

## To free a block:
- coalesce and place on appropriate list (optional)

## Tradeoffs
- faster search than sequential fits (i.e., log time for power of two size classes)
- controls fragmentation of simple segregated storage
- coalescing can increase search times
  - deferred coalescing can help

# Buddy systems

## Special case of segregated fits.

- all blocks are power of two sizes

## Basic idea:

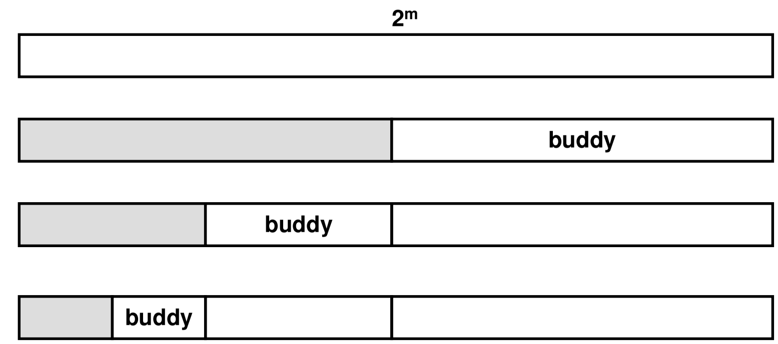
- Heap is $2^m$ words
- Maintain separate free lists of each size $2^k$, 0 <= k <= m.
- Requested block sizes are rounded up to nearest power of 2.
- Originally, one free block of size $2^m$.



# Buddy systems (cont)

## To allocate a block of size $2^k$:

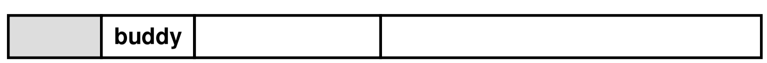
- Find first available block of size $2^j$ s.t. k <= j <= m.
- if j == k then done.
- otherwise recursively split block until j == k.





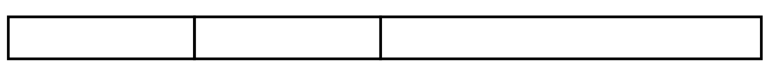

# Buddy systems (cont)

## To free a block of size $2^k$:




## To free a block of size $2^k$:

- if buddy free, coalesce with buddy and return new block to free list



## To free a block of size $2^k$:

- if buddy not free, just return block to free list



# Buddy systems (cont)

## Key fact about buddy systems:

- given the address and size of a block, it is easy to compute the address of its buddy
- e.g., block of size 32 with address *xxx...x00000* has buddy *xxx...x10000*

## Tradeoffs:

- fast search and coalesce
- subject to internal fragmentation

# Internal fragmentation

**Internal fragmentation is wasted space inside allocated blocks:**

- **minimum block size larger than requested amount**
  - e.g., due to minimum free block size, free list overhead
- **policy decision not to split blocks**
  - e.g., buddy system
  - Much easier to define and measure than external fragmentation.

**Source of interesting computer science forensic techniques in the context of disk blocks**

- **contents of "slack" at the end of the last sector of a file contain directory entries.**
- **provide a snapshop of the system that copied the file.**



# Steele mail fraud case

**March 6, 1993 (Pittsburgh, PA)**

- **Phil McCalister, disgruntled associate at Pgh law firm Steele & Hoffman, after watching the movie "The Firm", copies school board billing records from firm's laptops onto some diskettes, then resigns.**

**July 29, 1993**

- **McCalister hands over 4 diskettes to postal instpectors as evidence of systematic overbilling of school systems by Charlie Steele, managing partner of Steele & Hoffman.**

**September, 1996**

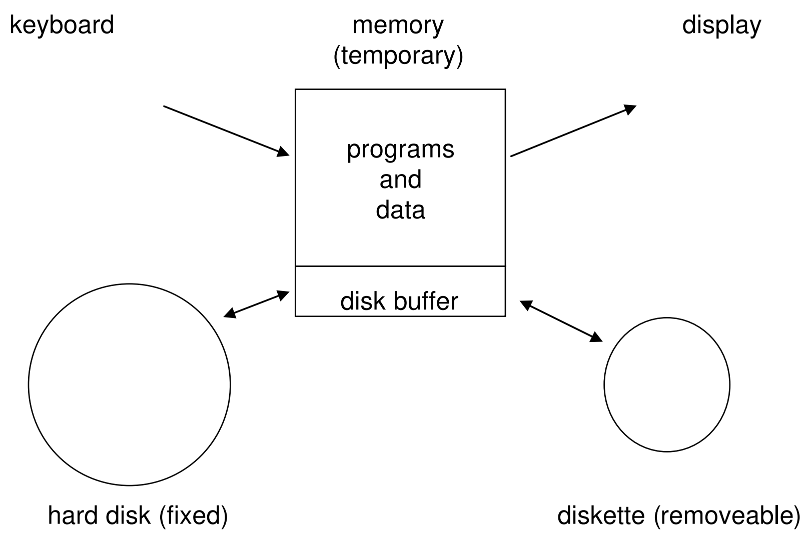
- **I'm asked by defense to determine if the 4 diskettes are the originals from March 6, 1993 (they weren't).**

**December, 1996**

- **Despite brilliant testimony by the computer expert witness, Charlie Steele convicted of mail fraud and sentenced to 3 years in federal pen and $80,000 fine.**

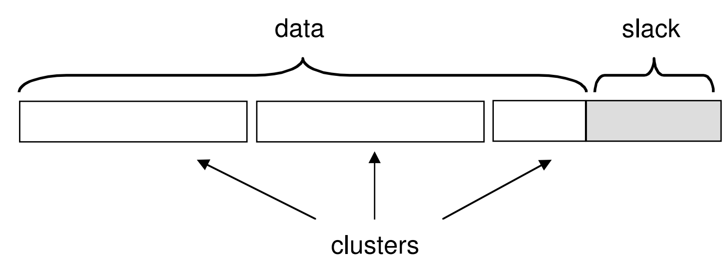


# Anatomy of a Computer



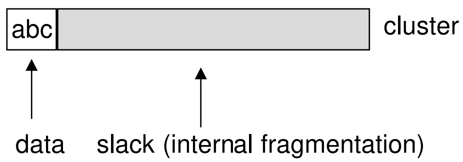



# How a DOS file is organized into clusters



# Internal fragmentation in DOS files

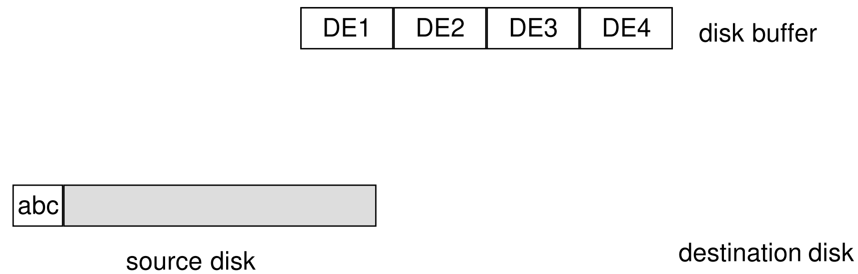
Files allocated in fixed size logical sectors






# How slack takes a picture of a disk when a file is copied (1)

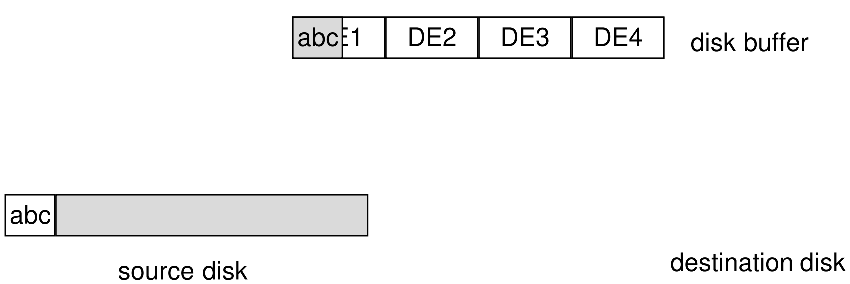
1. read source directory ("DE" is directory entry)






# How slack takes a picture of a disk when a file is copied (2)

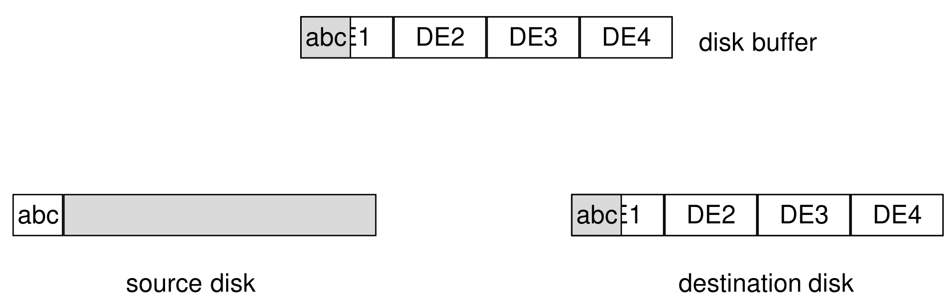
2. read file into disk buffer (notice that old slack is not copied into disk buffer!)






# How slack takes a picture of a disk when a file is copied (3)

3. write file to destination disk. Notice that slack now contains a snapshot of the files on the source disk when the file was copied.



# Federal diskette F1 is not a duplicate

```
Cluster 1,789, Sector 1,820  [F1:1991-$.IN C1638-1789]
Name     .Ext     Size      Date      Time    Cluster Arc R/O Sys Hid Dir Vol
-----------------------------------------------------------------------------
  ...     YS      33430  11-11-91   5:00 am       2      R/O Sys Hid
MSDOS    SYS      37394  11-11-91   5:00 am    5419      R/O Sys Hid
CONFIG   SYS         57  10-26-92   8:47 am    8998  Arc
AUTOEXEC BAT         24  10-26-92   8:47 am    8997  Arc
DOS                   0   3-22-93   4:40 pm      19                  Dir
WININST               0   3-22-93   4:41 pm     597                  Dir
WINDOWS               0   3-22-93   4:43 pm    3042                  Dir
COMMAND  COM      47845  11-11-91   5:00 am    5429  Arc
SCAN                  0   3-22-93   4:50 pm    5570                  Dir
WINA20   386       9349  11-11-91   5:00 am      14
HARCHLRD REG       1492   6-14-93  12:50 pm    5859  Arc
ASP                   0   3-23-93  11:59 am    6242                  Dir
DO                    0   3-23-93  12:01 pm    6295                  Dir
GOLF                  0   3-23-93  12:01 pm    6361                  Dir
LOTUS                 0   5-07-93   4:32 pm    5341                  Dir
NORTON                0   3-23-93  12:04 pm    6977                  Dir
```

**Source: Norton Utilities Diskedit program**



# Federal diskette F2 is not a duplicate

```
Cluster 501, Sector 532   [F2:CRIMALDI C498-501]
Name     .Ext     Size      Date      Time    Cluster Arc R/O Sys Hid Dir Vol
-----------------------------------------------------------------------------
...
WP51                  0   3-23-93  12:05 pm     7242                 Dir
XTALK                 0   3-23-93  12:13 pm     8910                 Dir
KATHY    REL       2239   6-14-93   1:20 pm     5869  Arc
FRECOVER DAT     101376   3-24-93  11:29 am     8951  Arc R/O
GO       BAT        198  10-26-92   8:47 am     8966  Arc
MENU     BAT        947  10-26-92   8:47 am     8967  Arc
SD       INI       2497  10-26-92   8:47 am     8968  Arc
XMENU    EXE       5521  10-26-92   8:47 am     8969  Arc
XMENU    PIF        296  10-26-92   8:47 am     8971  Arc
FRECOVER IDX         29   3-24-93  11:29 am    41442  Arc R/O Sys Hid
?UMMINGS           4763   5-20-93   2:45 pm     6617  Arc
?UMMINGS BK!       4664   5-19-93   8:18 pm     5895  Arc
```

**Source: Norton Utilities Diskedit program**



# Federal diskette F3 is not a duplicate

```
Cluster 936, Sector 967   [F3:SUMMARY C905-936]
Name     .Ext     Size      Date      Time    Cluster Arc R/O Sys Hid Dir Vol
-----------------------------------------------------------------------------
...
LOVIS                 0   4-09-93   2:23 pm    5902                  Dir
BACKUP             5825   4-23-93  10:10 am    5568  Arc
BURRELL               0   4-19-93  10:52 am    5380                  Dir
MIFFLINB              0   4-19-93  10:52 am    5382                  Dir
```

**Source: Norton Utilities Diskedit program**



# Files on the federal diskettes came from the hard disk

```
Cluster 2,023, Sector 2,054 [F2:PRICE C2020-2023]
Name     .Ext     Size      Date      Time    Cluster Arc R/O Sys Hid Dir Vol
DECEMBER 92       13003  12-14-92   2:07 pm     9215  Arc
EMWAUG17 92       10818   9-01-92   7:04 am     9219  Arc
EMWAUG3  92        9445   8-23-92   2:20 pm     9222  Arc
EMWDEC1  92        4453  12-07-92   1:31 pm     9225  Arc
EMWDEC14 92        9742   1-04-93  10:51 am     9227  Arc
EMWDEC7  92        4273  12-15-92  10:30 am     9230  Arc
EMWFEB1  93       15118   3-01-93   8:55 am     9232  Arc
EMWJAN4  93       21241   1-29-93   5:22 pm     9236  Arc
EMWJUL13 92        5706   7-31-92   9:24 am     9242  Arc
EMWJUL20 92        4331   7-31-92  10:34 am     9244  Arc
EMWJUL27 92        5822   8-23-92   2:14 pm     9246  Arc
```

**Source: Norton Utilities Diskedit program**

# The federal diskettes were produced in the afternoon of July 29, 1993

```
Cluster 1,069, Sector 1,100 [F2:HOGAN C1063-1069]
Name     .Ext      Size       Date       Time    Cluster Arc R/O Sys Hid Dir Vol
-----------------------------------------------------------------------------
PRESNT-1 WPG        4123   9-26-90  12:00 pm     8717  Arc
PRINTR-3 WPG        1899   9-26-90  12:00 pm     8719  Arc
SCALE    WPG        3071   9-26-90  12:00 pm     8720  Arc
STAR-5   WPG         391   9-26-90  12:00 pm     8721  Arc
TELPHONE WPG        6101   9-26-90  12:00 pm     8722  Arc
TROPHY   WPG        3891   9-26-90  12:00 pm     8724  Arc
PLEADS   PER        4903   3-07-93  11:07 am     8725  Arc
NEGOTIAT               0   3-23-93  12:07 pm     8727                    Dir
D                   3717   4-23-92   2:24 pm     8901  Arc
WP{WP}   SET        2496   7-29-93   2:10 pm     8902  Arc
COMMAND  COM       47845  11-11-91   5:00 am     8973  Arc
WP}WP{   CHK           0   7-29-93   1:54 pm        0  Arc
WP}WP{   TV1           0   7-29-93   1:54 pm        0  Arc
```

**Source: Norton Utilities Diskedit program**

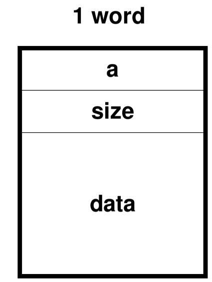


# Implementation Issues (cont)

## The simplest allocator

- allocate time: linear in total number of blocks
- free time: linear in total number of blocks
- minimum block size: two words

Format of allocated and free blocks




a = 1: allocated block
a = 0: free block

size: block size

data: application data
(allocated blocks only)

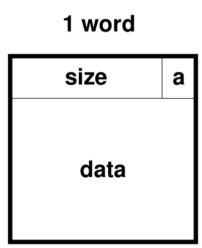


# Implementation issues (cont)

## A simple space optimization:

- exploit unused lower order size bits
- block size always a multiple of the wordsize
- reduces minimum block size from 2 words to 1 word

Format of allocated and free blocks



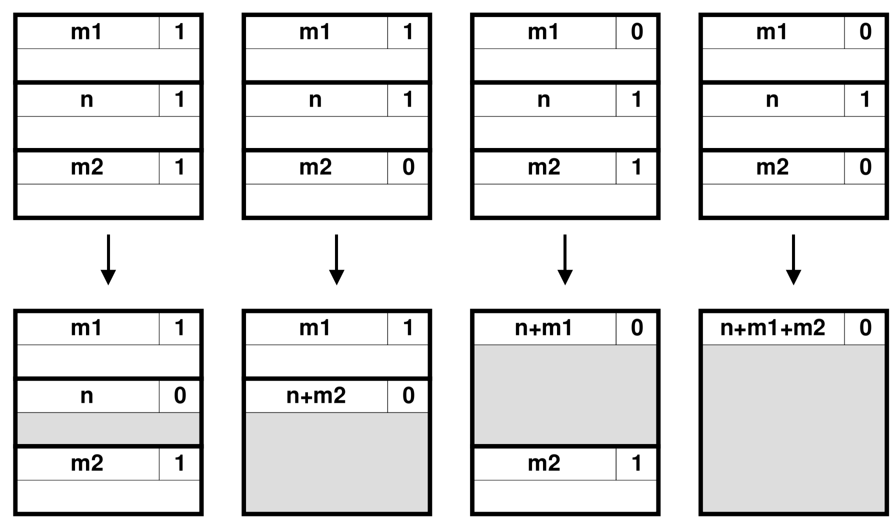

a = 1: allocated block
a = 0: free block

size: block size

data: application data
(allocated blocks only)



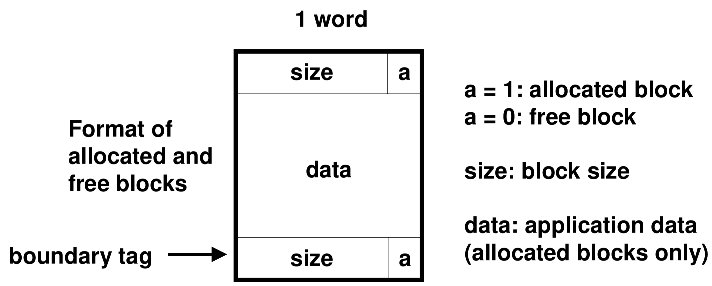
# General coalescing



# Implementation issues

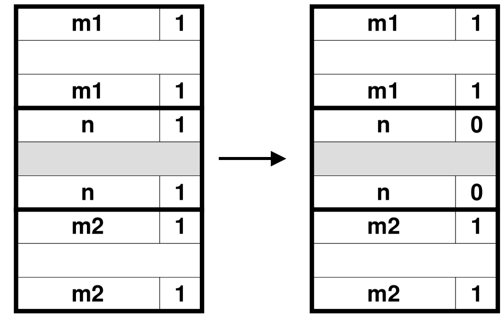
## Boundary tags [Knuth73]

- replicate size/allocated word at bottom of free blocks
- allocate time: linear in total number of blocks
- free time: constant time (with general coalescing)
- minimum block size: 2 words



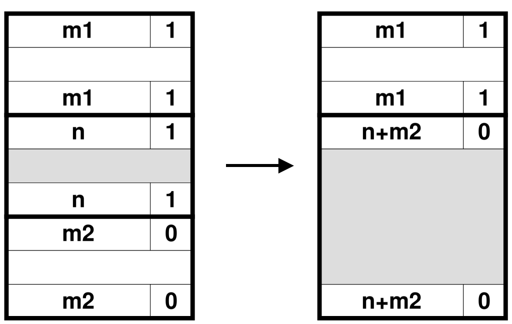



# Constant time coalescing (1)



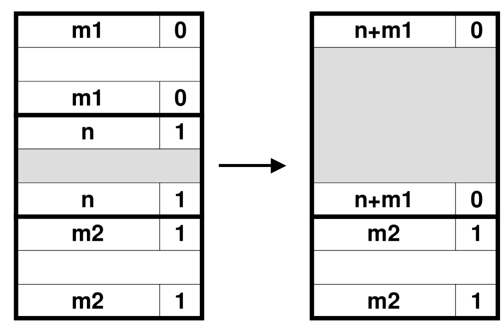



# Constant time coalescing (2)

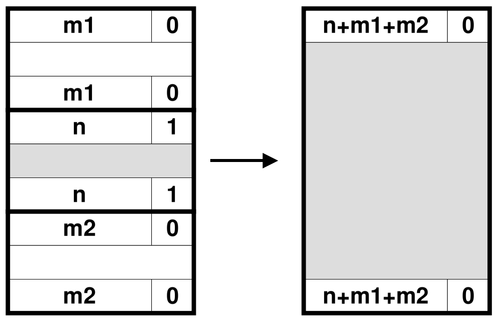





# Constant time coalescing (3)



# Constant time coalescing (4)



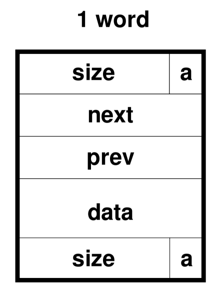



# Implementation issues (cont)

## Embedded free list

- embed link fields in body of each free block
- allocate time: linear in total number of *free* blocks
- free time: constant time (with general coelescing)
- minimum block size: 4 words

Format of
allocated and
free blocks




a = 1: allocated block
a = 0: free block

size: block size

next: points to next block in free list
prev: points to previous block in free list

data: application data
(allocated blocks only)



# Food for thought

**Can we avoid having two conditionals in the inner loop of the free block list traversal**

- one conditional to check size
- one conditional to check that the entire list has been searched

**Can we implement a sequential fits mechanism with constant time coalescing and a minimum block size of three words instead of four words?**



# For more information

**D. Knuth, "The Art of Computer Programming, Second Edition, Vol I, Fundamental Algorithms", Addison Wesley, 1973**

- the classic reference on dynamic storage allocation

**Wilson et al, "Dynamic Storage Allocation: A Survey and Critical Review", Proc. 1995 Int'l Workshop on Memory Management, Kinross, Scotland, Sept, 1995.**

- comprehensive survey
- /afs/cs/academic/class/15-213/doc/dsa.ps