



TEL AVIV UNIVERSITY

# מערכות הפעלה

ערן טרומר  
סמסטר א' תשע"ב

## הרצאה 3

קלט/פלט (המשך)

# מנהלי התקן (device drivers)

- ❖ מנהל התקן הוא מודול תוכנה עם ממשק סטנדרטי ששולט על התקן חומרה מסוים, כמו בקר דיסקים מסוג מסוים או בקר רשת מסוג מסוים
- ❖ השגרה שמופעלת על ידי קריאת מערכת קוראת למנהל ההתקן המתאים
- ❖ לעיתים מערכת ההפעלה קוראת למנהל התקן באופן לא ישיר: תוכנית קוראת מקובץ ומערכת ההפעלה קוראת למנהל ההתקן של הדיסק על מנת לחפש ולקרוא את הנתונים
- ❖ לא כל מנהל התקן שולט על חומרה; לפעמים מנהל התקן שולט על התקן וירטואלי, למשל מערך דיסקים

# ממשק של מנהל התקן כקובץ (מקרה אופייני בלינוקס)

- ❖ שגרת פסיקה שמופעלת כאשר ההתקן מפעיל פסיקה  
(interrupt handler / interrupt service routine)
- ❖ שגרה לאתחול ההתקן (initialize)
- ❖ שגרה להתחברות (open) – זמינות, הרשאות והשגת משאבים
- ❖ שגרה להתנתקות (close) – סיום משימות ושחרור משאבים
- ❖ שגרות לקריאת וכתיבת נתונים (write, read)
- ❖ שגרה להזזת מצביע הקלט/פלט (seek) עבור התקני גישה ישירה  
(random access)
- ❖ העברת פקודות מיוחדות להתקן או למנהל (ioctl)
  - דוגמה: קצב שידור של serial port

ממשק אחר למנהלי התקן של בקרי רשת תקשורת ושל בקרי תצוגה גרפית.

# דוגמה: סיפורה של גישה לדיסק

- ❖ תוכנית מבצעת קריאת מערכת – קריאה מקובץ (שכבר נפתח)
- ❖ מערכת ההפעלה מפעילה את שגרת read של מנהל ההתקן
  - מנהל ההתקן שולח הוראת בערוץ פלט אל ההתקן: "קרא N בלוקים ממקום M וכתוב את תוכנם לזיכרון בכתובת A"
- ❖ ... המתנה (אולי פעילה) ...
- ❖ ההתקן קורא מידע מהמדיה המגנטית ושולח אותו בגישה ישירה לזכרון
- ❖ ההתקן יוצר פסיקת חומרה
- ❖ המעבד קופץ לשגרת הפסיקה של מערכת ההפעלה
  - מערכת ההפעלה מריצה את שגרת הפסיקה של מנהל ההתקן
    - מנהל ההתקן מעתיק את החלק הרלוונטי של המידע בכתובת A אל הזיכרון של התוכנית
    - מנהל ההתקן מדווח על סיום פעולת הקריאה
- ❖ התוכנית ממשיכה לרוץ

# הטמנה וחציצה

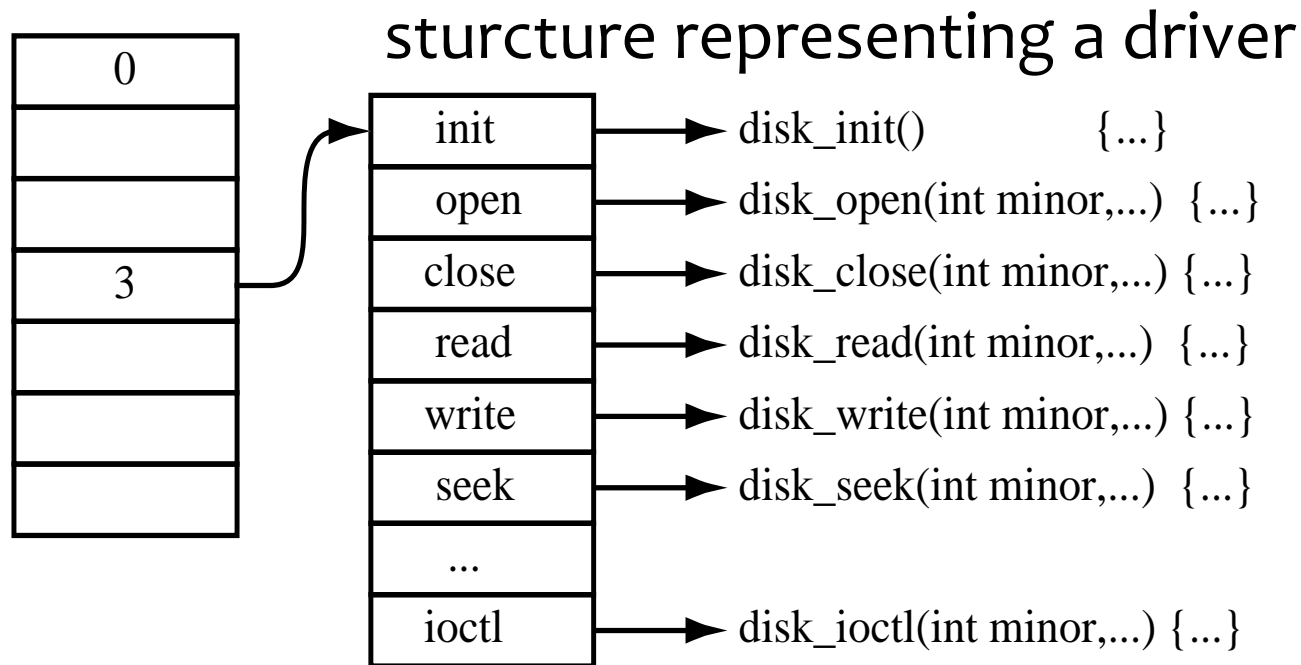
- ❖ **הטמנה (caching):** מערכת ההפעלה שומרת בזיכרון הראשי בלוקים של נתונים שנקראו לאחרונה מהתקני זיכרון חיצוניים (דיסקים)
- ❖ גישה חוזרת לבלוק מחזירה את תוכנו מהמטמון ללא גישה להתקן החיצוני האיטי
- ❖ מערכת ההפעלה תקרא מידע נוסף מיוזמתה (read-ahead)
- ❖ **חציצה (buffering):** מערכת ההפעלה שומרת בזיכרון הראשי בלוקים שמיועדים להיכתב להתקני זיכרון חיצוניים ומעכבת את הכתיבתם להתקן
- ❖ כתיבה חוזרת לאותו בלוק משנה את תוכן הבלוק בחוצץ; רק הנתונים המאוחרים יכתבו להתקן החיצוני
- ❖ חציצה (עיכוב כתיבה) חוסכת גישות שמשכתבות נתונים ומאפשרת לכתוב להתקן כאשר הוא פנוי

# מנהלי התקן עם וללא הטמנה

- ❖ מערכת ההפעלה מנהלת מאגר חוצצים מרכזי (מטמון) עבור מנהלי התקן של התקני זיכרון חיצוניים
- ❖ התקנים שאינם התקני זיכרון (מקלדת, עכבר, מדפסת, רשת תקשורת) אינם זקוקים להטמנה, אבל חלקם זקוקים לחציצה
- ❖ חציצה מאפשרת לקרוא נתונים שמגיעים מההתקן (מקלדת למשל) גם אם תוכנית אינה מבצעת קריאה באותו רגע ממש, ולכתוב נתונים בקצב שמתאים להתקן (מדפסת, מודם)
- ❖ בלינוקס/יוניקס יש שתי טבלאות של מנהלי התקן: עם הטמנה (block) וללא הטמנה (character), חוץ ממנהלי התקן לבקרי רשת ולבקרי תצוגה גרפית

# התייחסות להתקנים ביוניקס

block devices



❖ התקן מיוצג על ידי שני מספרים: ראשוני שמייצג את מנהל ההתקן

שמטפל בהתקן ומשני שמייצג את ההתקן עבור המנהל

❖ תוכניות מתייחסות להתקן על ידי גישה לקובץ מיוחד שמכיל מזהה טבלת

מנהלים (עם/ללא הטמנה) ואת שני המספרים

❖ ניתן להתייחס להתקן כאל קובץ, להתחבר אליו ולקרוא/לכתוב נתונים

# מנהלי התקן בלינוקס/יוניקס

❖ מנהל ההתקן קיים וההתקן עצמו קיים:

```
# ls -l /dev/lp0  
crw-rw---- 1 root daemon 6,0 may 5, 1988 /dev/lp0  
# cat < /dev/lp0  
skfjkl...
```

❖ מנהל ההתקן קיים, אבל אין התקן פיזי:

```
# cat < /dev/lp0  
cat: -: Input/output error
```

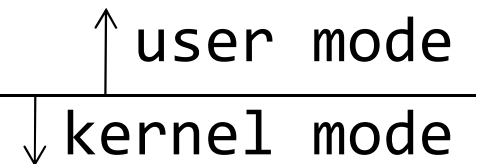
❖ הקובץ המיוחד קיים, אבל אין מנהל התקן בטבלה:

```
# cat < /dev/lp0  
bash: /dev/lp0: No such device
```



# מסלול השגרות במערכת ההפעלה

```
...  
fd = open("/dev/lp0",...); // returns 4  
read(fd, ...);
```



file descriptor 4 for process 1234 → special file (6,0)

read system call:

```
call character_devices[6].read(0,...)
```

device driver for lp:

```
int read(...) { retrieve bytes from buffer }
```

```
void isr(void) { read bytes from controller to buffer }
```

# דוגמאות נוספות לקבצים מיוחדים והתקנים

```
# ls -ld /dev/*
```

```
brw-rw---- 1 root root      11,   0 Nov  7 18:47 /dev/cdrom
crw-r----- 1 root kmem     10, 144 Nov  7 18:47 /dev/nvram
brw-rw---- 1 root disk       8,   0 Nov  7 18:47 /dev/sda
crw--w---- 1 root tty        4,   0 Nov  7 18:47 /dev/tty0
crw--w---- 1 root tty        4,   1 Nov  7 18:47 /dev/tty1
crw-rw---- 1 root root     188,   0 Nov  7 18:47 /dev/ttyUSB0
crw-rw---- 1 root audio     14,   4 Nov  9 18:47 /dev/audio
brw-rw---- 1 root disk    253,   0 Nov  9 18:47 /dev/dm-0
crw-rw-rw- 1 root root       1,   5 Nov  7 18:47 /dev/zero
crw-rw-rw- 1 root root       1,   8 Nov  7 18:47 /dev/random
```

(partial list)

# התאמת מנהל התקן להתקן

❖ התקנה ידנית

❖ חיבור אוטומטי

- בחיבורי התקנים מודרניים (לדוגמה USB) יש לכל דגם התקן מזהה ייחודי

- מנהל ההתקן מספר למערכת ההפעלה באיזה מזהים הוא תומך

- כאשר מחברים התקן חדש הנמצא ברשימה, מערכת ההפעלה מפעילה את מנהל ההתקן המתאים

❖ חיפוש אוטומטי

- מערכת ההפעלה מבקשת מתוכנית ייעודית לחפש מנהל התקן באינטרנט על פי המזהה

❖ ההתקן מספק מנהל התקן דרך ממשק סטנדרטי

# יצירת הקבצים המיוחדים בלינוקס

- ❖ הקבצים המיוחדים ב-`/dev`, המובילים אל טבלאות מנהל ההתקן, הם הממשק מולו עובדות רוב התוכניות
- ❖ הקבצים עצמם נוצרים על ידי תוכנית שירות המהווה חלק ממערכת ההפעלה במובן הרחב (לא בליבה)
- ❖ בעבר: סקריפט בעליית המערכת יצר את כל הקבצים האפשריים
- ❖ כיום: מנגנון מסובך (`udev`) עם אוסף כללים: "אם מנהל התקן X מצא התקן של יצרן Y מדגם Z אז צור קובץ מיוחד F אשר מצביע אליו"
- ❖ לתוכניות רגילות לא אכפת

# התקנים כקבצים – מדוע?

❖ ממשק פשוט וסטנדרטי

❖ ניתן להפעיל פקודות קבצים על התקנים

❖ יצירת עותק מושלם של CD לתוך קובץ ISO:

```
cat /dev/cdrom > image.iso
```

❖ דחיסה ושליחה של דיסק קשיח שלם לשרת ברשת:

```
cat /dev/sdb | gzip -c | ssh user@server 'cat > backup'
```

❖ השמעת רעש לבן:

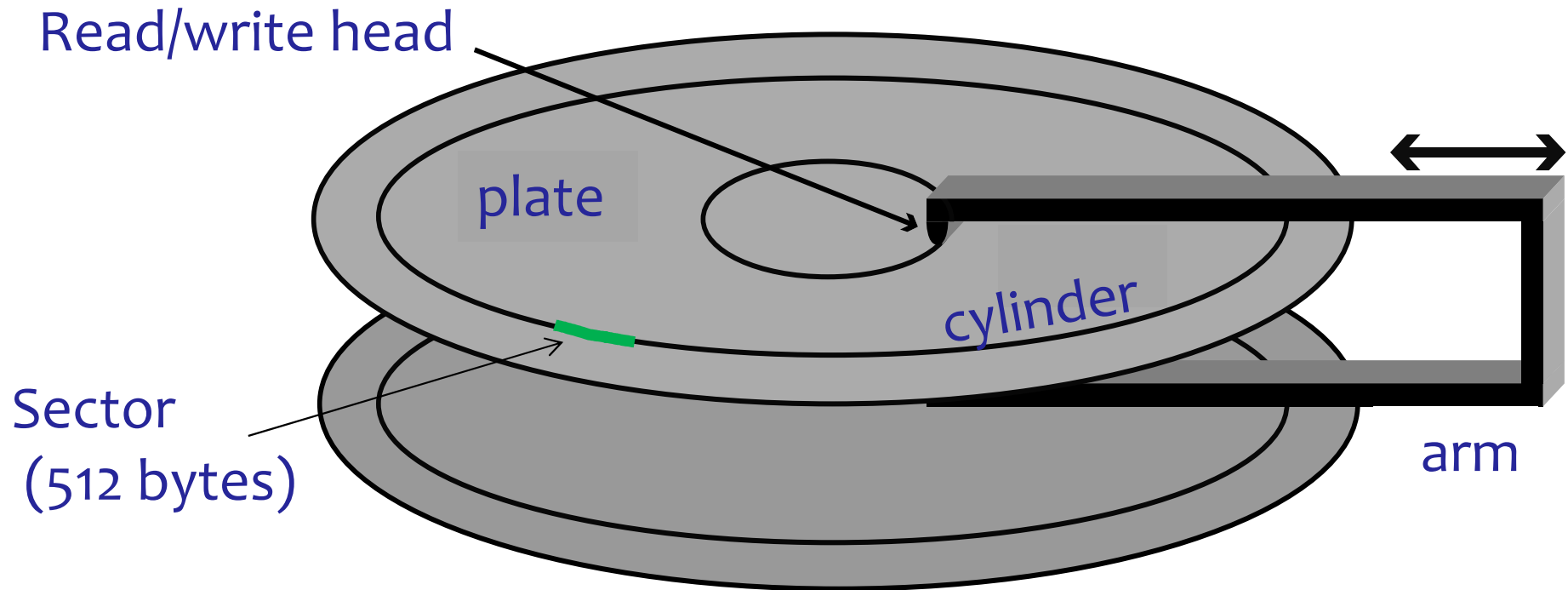
```
cat /dev/random > /dev/audio
```

❖ "מתכנת אמיתי" מריץ

```
cat /dev/audio > program
```

ואז שורק.

# דיסקים קשיחים (hard disks)



- ❖ משמשים לאחסון קבצים ונהרחבה של הזיכרון הפיזי ולכן ביצועיהם משפיעים ישירות על ביצועי מערכת המחשב כולה
- ❖ מסתובבים במהירות קבועה של 3600-15000 סיבובים בדקה
- ❖ קיבולת של עד 2 TB, קצבי העברה אפקטיביים של 100 MB/s
- ❖ המספרים נכונים לשנת 2011; זמן סיבוב ורוחב פס משתפרים לאט

# תזמון דיסקים קשיחים

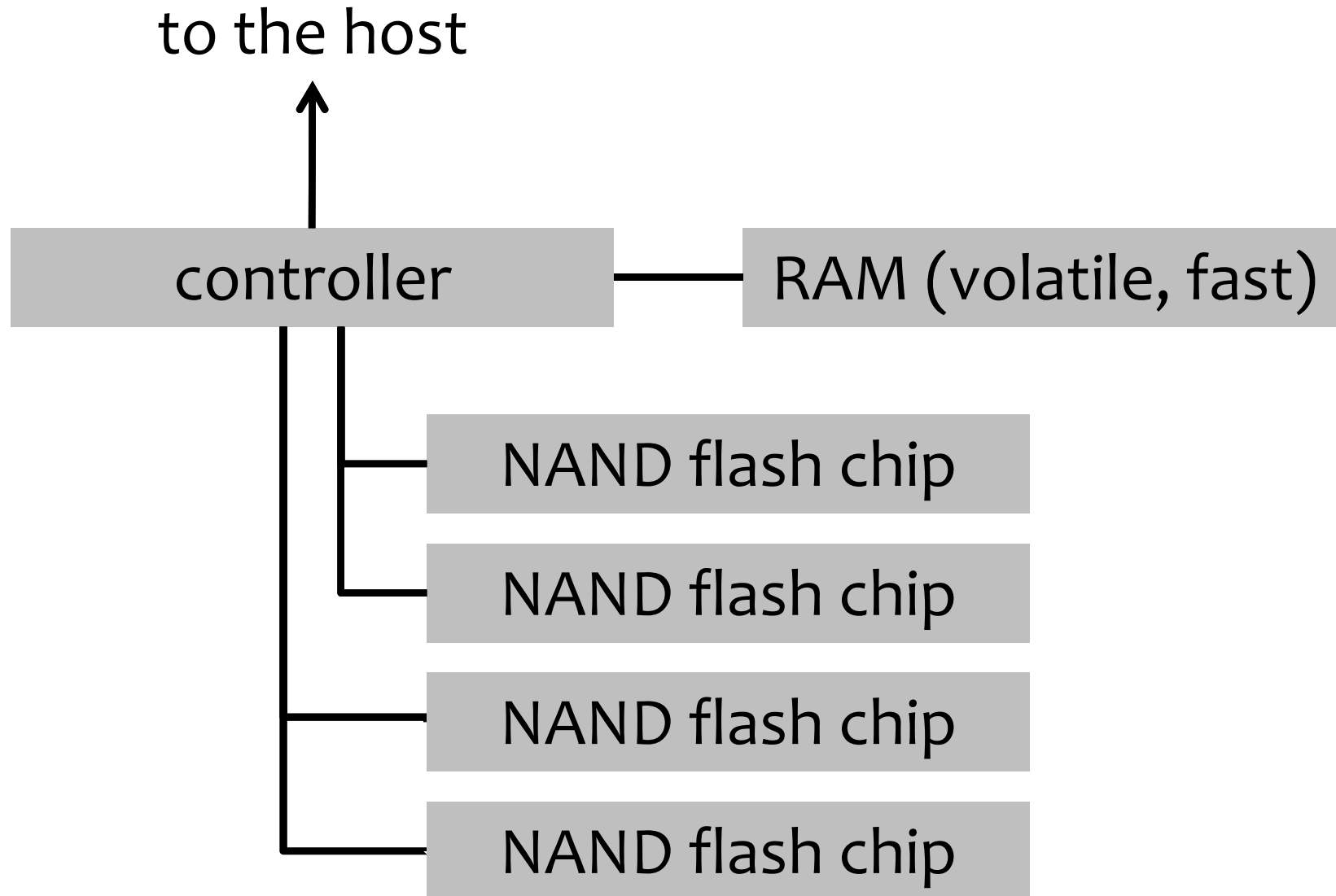
- ❖ בקשות גישה מצטברות בתור (מתהליכים שונים וגישות גדולות)
- ❖ איזה בקשה לבצע כעת? שאלת מדיניות
- ❖ ביצוע לפי סדר קבלה (FCFS): הוגן לחלוטין אבל לא יעיל; הזרוע עלולה לנוע שוב ושוב מצד לצד על מנת לקרוא קטע (סקטור) בודד בכל פעם
- ❖ מזעור זמן השיוט (SSTF): הבקשה הבאה שתשורת היא הבקשה שניתן להגיע אליה תוך פרק הזמן המזערי; לא הוגן עד כדי הרעבה
- ❖ מדיניות מעלית: הזרוע נעה מהמסילה הפנימית לחיצונית וחזרה ומשרתת את כל בקשות הגישה לפי סדר תנועת הזרוע, בדומה לאוטובוס עם מסלול קבוע; הוגן ויעיל

# ואריאציות על מדיניות מעלית

- ❖ scan: הזרוע נעה פנימה והחוצה בין שתי המסילות הקיצוניות ומשרתת בקשות שהיא פוגשת בדרך
- ❖ c-scan: כנ"ל, אבל משרתים בקשות רק בדרך פנימה, את הדרך החוצה עושים במהירות; יותר הוגן ואולי מעט פחות יעיל (מעט כי הזרוע מאיצה)
- ❖ look: כמו scan אבל הזרוע לא נעה מעבר למסילה הקיצונית שיש עבודה בקשה. יותר יעיל ופחות הוגן
- ❖ c-look: מובן מאליו
- ❖ אפשר להפעיל את מדיניות דומה גם לגבי קטעים בתוך מסילה
- ❖ האם כדאי לשרת בקשות למסילה שנמצאים בה שנכנסו לתור לאחר שהזרוע הגיע למסילה. מדוע? שקלו הגינות, יעילות

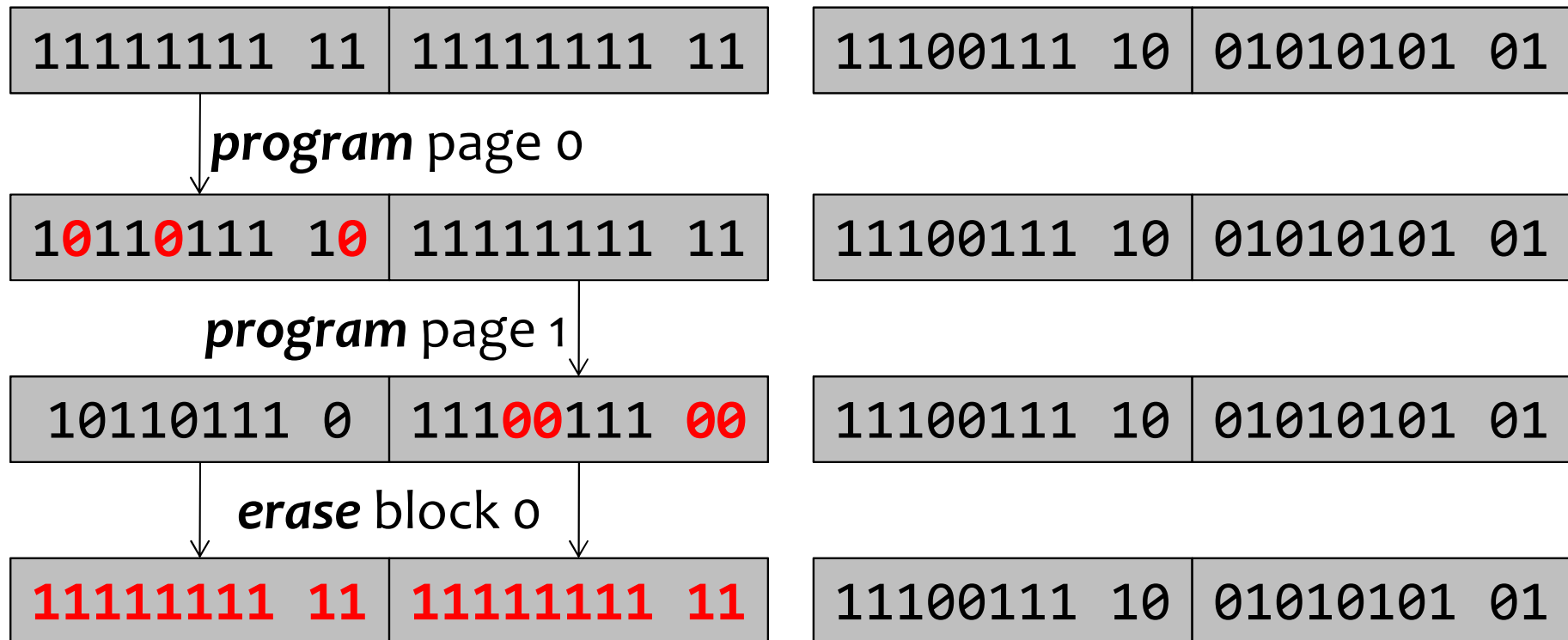


# אחסון מבוסס flash (Solid State Disks)



# זיכרונות NAND Flash

❖ הזיכרון מחולק לבלוקים, כל בלוק מחולק ל-128 או 64 דפים, כל דף מכיל מספר סקטורים של נתונים וגם נתונים אודות הנתונים (metadata)



# SSDs צריכים בקר חכם

- ❖ לא כדאי לשכתב סקטור במקום, כי צריך יהיה למחוק בלוק שלם
- ❖ תאי זיכרון נשחקים (נהרסים) במהלך כתיבות/מחיקות; מספר מחזורי הכתיבה הוא בין אלף ל-100 אלף (הולך ונהיה גרוע)
- ❖ הבקר כותב את סקטור i לדף j וזוכר את המיפוי
- ❖ מבנה הנתונים של המיפוי נשמר ב-RAM של הבקר ועל ה-flash
- ❖ הנתונים אודות הנתונים מכילים את זהות הסקטורים השמורים בדף, קוד לתיקון שגיאות, ומידע אחר שהבקר צריך

# דוגמה לתכנון בקר של SSD

❖ SSD בגודל 32GB שיש לו 32K בלוקים בגודל 1MB, בכל אחד 128 דפים של 8K, ויש לו 1MB של RAM

❖ איך לנהל את המיפוי?

- כל 1MB של סקטורים רצופים ימופו לבלוק אחד ב-flash
- את המיפוי הזה ה-SSD ישמור ב-RAM שלו
- אלגוריתם (פשוט) למצוא סקטור נתון בזמן קריאה ולשכתב סקטור
- איזון שחיקה: מחיקת כל הבלוקים באותו קצב בערך
- המיפוי נכתב ל-flash לפני שמכבים את ה-SSD; לא הנחה כל כך טובה כי לפעמים אספקת החשמל נפסקת בפתאומיות
- שיפורים מבוססים על עקרונות של מערכות קבצים מתאוששות ועל log-structured file systems שנדון בהם בהמשך

# אמינות במערכות גדולות

- ❖ נניח שלדיסק בודד יש הסתברות של 0.1% להתקלקל במהלך 24 שעות
- ❖ (בפועל ההסתברות ליחידת זמן גדולה יותר עבור תינקות וזקנים)
- ❖ תוחלת חיי הדיסק בערך 3 שנים (התפלגות גיאומטרית)
- ❖ אם כמות הנתונים דורשת שימוש ב-10 דיסקים (אחד לא מספיק), אז תוחלת הזמן לתקלה הראשונה היא \_\_\_\_\_

# אמינות במערכות גדולות

- ❖ נניח שלדיסק בודד יש הסתברות של 0.1% להתקלקל במהלך 24 שעות
- ❖ (בפועל ההסתברות ליחידת זמן גדולה יותר עבור תינוקות וזקנים)
- ❖ תוחלת חיי הדיסק בערך 3 שנים (התפלגות גיאומטרית)
- ❖ אם כמות הנתונים דורשת שימוש ב-10 דיסקים (אחד לא מספיק), אז תוחלת הזמן לתקלה הראשונה היא 100 ימים; מעט מדי
- ❖ אין מה לחשוב אפילו על מערך של 100 או 1000 דיסקים...

# גילוי ותיקון שגיאות

❖ אריתמטיקה בינרית:  $0=0+0$ ,  $1=0+1$ ,  $1=1+0$ ,  $0=1+1$  (xor)

❖ נגן על 4 סיביות נתונים  $x_1, x_2, x_3, x_4$  על ידי שימוש בסיבית

נוספת  $x_5$  ואילוץ אלגברי  $x_1 + x_2 + x_3 + x_4 + x_5 = 0$

❖ חשב את  $x_5$   $1 + 1 + 1 + 0 + \_ = 0$

❖ גילוי שגיאה  $1 + 1 + 0 + 1 + 1 = 0$

❖ תיקון (השלמת חוסר)  $1 + ? + 1 + 1 + 1 = 0$

❖ האם אפשר לגלות ככה שגיאה בשתי סיביות? או לגלות וגם לתקן שגיאה?

❖ גילוי ותיקון מתבצע ברמת הסקטור (וגם RAID, נדון בהמשך)

# מחיצות

- ❖ חלוקה של דיסק למספר דיסקים לוגיים על פי טבלה ששמורה בתחילת הדיסק
- ❖ כל מערכות ההפעלה יודעות לפענח את הטבלה
- ❖ משמשת להתקנת מספר מערכות הפעלה או מספר מערכות קבצים על אותו דיסק פיזי
- ❖ בשימוש בעיקר במחשבים אישיים



# דיסקים לוגיים (Logical Volumes)

- ❖ דיסק אחד או יותר מחולקים למקטעים (extents) בגודל קבוע
- ❖ ניתן להרכיב דיסק לוגי מכל קבוצה של מקטעים
- ❖ מאפשר להרכיב דיסקים לוגיים מהירים יחסית (למשל על ידי שימוש במסילות חיצוניות של מספר דיסקים)
  - מהירות הדיסק הלוגי יכולה להיות גדולה ממהירות כל אחד מהדיסקים הפיזיים: זמן תגובה וגם קצב העברה
- ❖ מאפשר להגדיל ולהקטין דיסק לוגי על ידי הוספת או גריעת מקטעים (מערכת הקבצים, מבנה הנתונים שעל הדיסק, צריכה לתמוך בהגדלת והקטנת הדיסק)
- ❖ בשימוש בעיקר ביוניקס ולינוקס

# מערכי דיסקים (RAID)

Redundant Array of Inexpensive Disks ❖

מספר דיסקים משמשים כהתקן אחסון אחד ❖

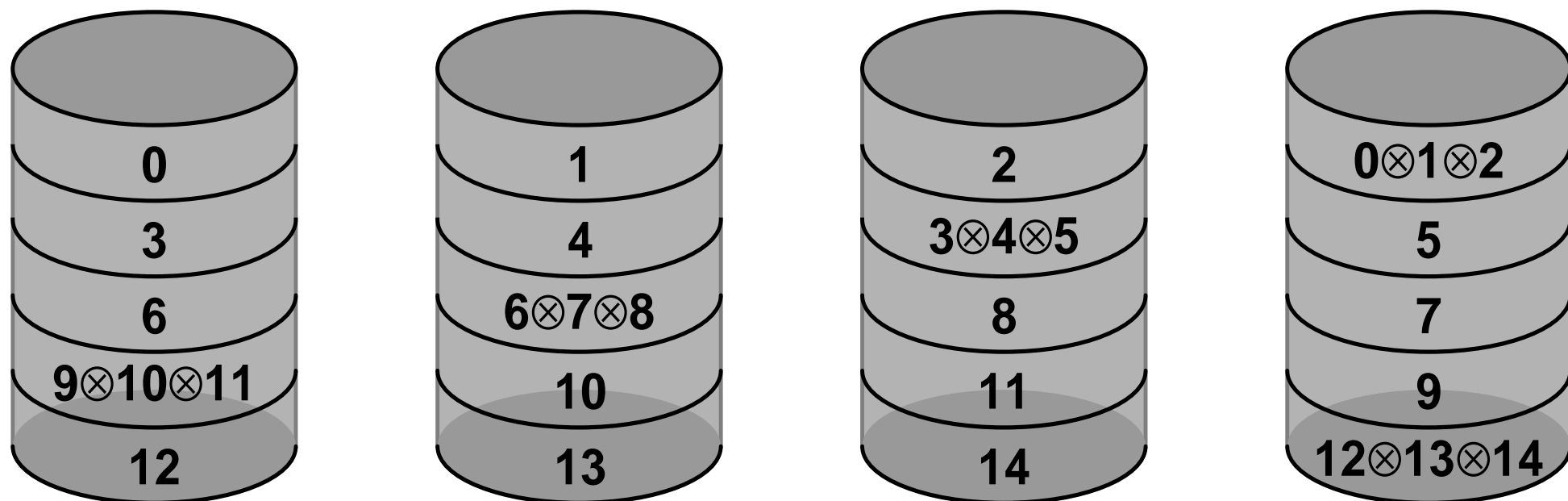
מבוסס על מיפוי בלוקים של נתונים בהתקן הוירטואלי (מערך ❖

הדיסקים) לדיסק+היסט, כלומר למיקום של בלוק פיזי של נתונים

מספר ואריאציות שנקראות רמות שמספקות קומבינציות שונות ❖

של שירותים; רמה גבוהה אינה בהכרח טובה יותר לכל שימוש

# שחזור נתונים עם RAID



- ❖ המערך מחולק לרצועות
- ❖ רצועה שומרת  $n-1$  בלוקים של נתונים ובלוק זוגיות
- ❖ אפשר לקרוא בלוק בודד; או לקרוא רצועה ולבדוק שגיאות
- ❖ אפשר לשחזר כל בלוק חסר, ואפשר גם לשחזר דיסק שלם שהתקלקל
- ❖ כתיבה של בלוק על ידי קריאה של הבלוק ובלוק הזוגיות ועדכון, או על ידי כתיבה של רצועה שלמה בלי לקרוא קודם כלום