**Fire system-מערכת קבצים**

מערכת הפעלה לא מטפלת במשתנים בודדים

מערכת הפעלה מטלפת במידע שמאורגן ביחידות שמכנים אותם קבצים.(קבצי נתונים)

המידע שמאורגן בקבצים נשמר בדיסקים, HARD DISC, CD, USB וכו'

אך האמצעי העיקרי עדיין נשאר DISC.

מבנה הדיסק הקשיח וחישוב הקיבולת שלו

דיסק קשיח בנוי ממשטחים (צלחות) אשר יושבות על ציר משותף. הציר מחובר למנוע חזק שמסובב את הצלחות האלה למהירות 7200 סל"ד.

המנוע מופעל אוטומטית עם הפעלת המערכת וכל פניה לדיסק מלווה בקריאה או כתיבה של אינפורמציה על הדיסק.

הצלחות של הדיסק מצופות בחומר מגנטי.

כל צלחת מכילה מסילות ברדיוס שונה.

על פני המסילות מתבצעת קריאה וכתיבה של המידע.

בשתי צידי הצלחות יש ראשי כתיבה קריאה.

ראשי קריאה כתיבה יושבים על זרוע משותף.

קריאה וכתיבה מתבצעות בו זמנית על מבנה שקוראים לו **צילינדר או גליל**,

המבנה הזה בנוי מאותן מסילות בעלות רדיוס זהה על כל הצלחות.

על כל מסילה ישנה חלוקה לסקטורים. משולשי פיצה.

סקטור מכיל אותו כמות מידע במסילות של רדיוס שונה, בד"כ זה חצי קילו-בייט.

לבצע חישוב של קיבולת הדיסק בעל שלושה משטחים כאשר על כל משטח יש 40 מסילות מצד אחד.

(40 מלמעלה 40 מלמטה)

וכל מסילה מכילה 20 גזרות. (סקטורים)

דיסק – 80 מסילות

מסילה – 20 גזרות

גודל סקטור – בד"כ חצי קילו-בייט. (0.5)

קיימים 3 דיסקים.

**חישוב:**

80x20=1600

1600x0.5=800

800x3=2400

**Cluster**

אחד המושגים החשובים לארגון קבצים על הדיסק נקרא CLUSTER.

CLUSTER זה בלוק נתונים (הקצאת זיכרון) שהוא בנוי כרשימה מקושרת והוא נראה כך:

כל קובץ מחולק לחלקים בתוך ה-CLUSTER.

Fire1:

0->1

1->2

2->3

…

59-<EOF

Fire2:

60->61

61->62

62->63

…

73->EOF

סיומת הקובץ מסתיימת בEOF- END OF FILE

כאשר רוצים לאכסן קובץ על הדיסק- מערכת ההפעלה מחפשת עבורו CLUSTER פנוי ומשלבת אותו בתוך ה-CLUSTER

במידה ו-CLUSTER לא פנוי, מערכת הפעלה לא נוגעת בו וממשיכה בחיפוש אחר מקום על הדיסק עבור הקובץ.

מי שאחראי על שילוב הקבצים על הדיסק זה FAT- FIRE ALLOCATION TABLE

בעזרת פניה ל-FAT , המערכת מוצאת מקום עבור הקובץ.

אם קובץ לא יכול להתאכסן ב-CLUSTER הקיים, מערכת ההפעלה מגדירה CLUSTER נוסף , מעדכנת את הטבלה ומשלבת את הקובץ על הדיסק.

דיסק מחולק ל-PARTITIONS

לכל PARTITION יש את ה-DIRECTORY שלו.

אם אנחנו רוצים לבצע מעקב אחרי שילוב הקבצים בדיסק, יש אפשרות לעיין בטבלה שקוראים לה DIRECTORY.

ב-UNIX לטבלת ה-DIRECTORY קוראים inode

ב-WINDOWS קוראים לה DIRECTORY.

בצילינדר 0 בסקטור 0 נמצא טבלת המחיצות – VTOC- VOLUME TABLE OF CONTENT

ה-15 סקטורים הבאים אחרי סקטור 0, הם מכילים את ה-BOOT BLOCK

BOOT BLOCK- קטע קוד שמבצע טעינה של מערכת הפעלה מהדיסק לזיכרון הראשי.

פקודות של מערכת הפעלה

קרנל

התוכניות של קרנל נטען לתוך הזיכרון הראשי- להפעלה שלהן צריך מעט מאוד זמן. גישה נוחה ומהירה.

קרנל של מערכת הפעלה מחולק לשני חלקים

אחד לא תלוי בחומרה- הוא מבצע ניהול קבצים תהליכים , ניהול זיכרון

אחד תלוי בחומרה – התקנים פיזיים, התקני חומרה, מנהל קלט פלט

בעת הטעינה של UNIX הן כאלה שהקרנל נטען לתוך הזיכרון הראשי

מערכת קבצים של UFS- UNIX FILE SYSTEM

UFS מתמקמת על הדיסק בצורה כזאת שהיא תופסת מקומות שמחולקים לסקטורים.

כל חלוקה מכילה 16 סקטורים.

סקטור 0, סקטור 1....עד 15.

אח"כ מסטור 16 עד 31 וכו'

**סקטור 0 –** מכיל את טבלת המחיצות. PARTITION TABLE - VOLUME TABLE OF CONTENTS

PARTITION TABLE – C D E

VOLUME TABLE OF CONTENTS

**בסקטורים 1 עד 15 מכיל BOOT.**

תוכנית הפעלה למערכת- מבצעת טעינה של מערכת הפעלה מהדיסק לזיכרון הראשי.

מהרגע שמופעל הBOOT מערכת ההפעלה מתחילה להיות בראשות המשתמש.

**ב16 סקטורים הבאים** נמצאים כל מיני פרטים על המערכת הפעלה שלנו – מספר בלוקים של נתונים , מספר צילינדרים, גודל של בלוק נתונים

שם של נקודת עגינה

נקודת עגינה זה שם של המחיצה \ עגינה ששם מתאכסנת התחלה של מערכת הפעלה ואנחנו צריכים את זה כדי שנוכל לבצע הפעלה של מערכת הפעלה.

**מ48 סקטורים ומעלה- תכנים של inode, טבלת ה- DIRECTORY**

כל הקבצים שיש לנו על הדיסק יהיו בסקטורים האלה

Inode יש שם קובץ , משקל, שם מורחב, תהליכי יצירה, בלוקים, מאפיינים.

**סוגי דיסקים-**

SSD – SOLID STATE DISC- הם לא בלתי מוגבלים.

SCSI- SMALL COMPUTER SYSTEM INTERFACE- הוא עובד יותר מהר עם עבודה עם דיסקים

CX- מספר בקר של החבילה

TX- TARGET – מספר דיסק SCSI. לכל חבילה יש את חבילת ה SCSI שלו.

DX- מספר הדיסק בתוך החבילה.

SX- SLICE – מספר PARTITION בתוך הדיסק.

C1 D2 E3 F4 H5

0C- בקר 0

1D- דיסק 1

5S- מתוך הדיסק, בוחרים את PARTITION H

נקודת עגינה של מערכת קבצים על המחשב נעשית ע"י פקודת MOUNT וזה נעשה באופן הבא:

Newfs/DEV/disk/c0t0d0s5 - להתקין מערכת קבצים חדשה על האמצעי הבא.

Mkdar/data- מחיצה – DATA

Mount/dev/disk/c0t0d0s5/data- נקודת עגינה

לביטול מערכת הקבצים (ביטול העגינה)

Umount/dev/disk/c0d0s5

Umount/data

שיטות גישה לקריאה מהקובץ (מהדיסק)

הקצאה רציפה

1.First fit - יתרון – מהירות חיסרון- זמן שמחפשים מקום

Best first- יתרון-השיטה הכי טובה להתאים לקובץ את מקומו ולחסוך מקום בדיסק חיסרון- זמן לחפש מקום

Worst fit- חיסרון- אין הקצאה אופטימלית

גישות לקובץ

Fcfs- first come first set- לוקחים בקשה ראשונה, מטפלים, עוברים לבקשה שניה וכו'.

מבצעים חיפוש קובץ לצורך הקריאה. אם לא מצאנו זה ממשיך עד END OF FILE.

הפעולת קריאה לא לוקחת זמן. היא עוברת על כל המשטחים עד המציאה.

Shortest seek first time-SSFT

סריקה של כל הרצף ולמצוא את הצילינדר הכי קרוב ולהתחיל לעבוד איתו.

מרוויחים זמן של פעולה פיזית של הזזת הזרוע. (מ-37 יעבור ל-45 ולא ל-207)

יתרון – חוסכים זמן

חיסרון- צריך לעבור על כל הרשימה של בקשות

SCAN

נעשית תנועה של ראש קריאה מהמקום הנוכחי לכיוון צילינדר האחרון כאשר הראש נע ממסילה למסילה, נעשית בדיקה אם המסילה נמצאת ברשימת הבקשות, אם כן, מתבצעת קריאה. אם לא, ממשיכים למסילה הבאה וכך עוברים על כל המסילות עד הצילינדר האחרון.

סריקה קדימה ואז יורד למטה.

צריך לדעת מאיזה נקודה הוא הגיע לנקודת קריאה כתיבה (53) אם הגיע מלמטה(47) המעלית תיהיה כלפי מעלה. אם הגיע מלמעלה (59) המעלית תהיה כלפי מטה.

שיטת SCAN לצורך איתור צילינדר שצריך להיכנס לעבודה מבוססת על כך שאם ברגע הזה עבדנו עם צילינדר 36 וצריך למצוא צילינדר שצריך לעבוד עליו- החיפוש נעשה מצילינדר 36, 35, 35 לכיוון 0. (מ36 להתחלת הדיסק) אם סרקנו ולא מצאנו, חוזרים לסוף של הדיסק ומשם מתחילים את החיפוש לכיוון התחלה.

סריקה מצילינדר הכי גבוה לכיוון 0.

הראשים זזים מ37 ל36 וכו'.

תרגיל לשיטות החיפוש של צילינדר הנתון לפי שלושת השיטות

התור של הבקשות הוא הבא- (בקשות קריאה מהדיסק)

98, 183, 37, 122, 14, 124, 65, 67

לבצע חישוב של זמני החיפוש לפי שיטות FCFS, FFST, SCAN

FCFS-

הזרוע מכוון ל-53. (ראש קריאה כתיבה)

14 37 53 65 67 98 122 124 183 הראש קריאה כתיבה נמצא ב53.

98 – 53 = 45

183-98=85

183-37= 146

122-37= 85

122-14= 108

124-14=110

124-65=59

67-65=2

45+85+146+75+108+110+59+2=640

SSFT- shortest seek first time

53 -> 65 -> 67 -> 37 -> 14 -> 98 -> 122 -> 124 -> 183

65-53=12

67-65=2

67-37=30

37-14=23

98-14=84

122-98=24

124-122=2

183-124=59

12+2+30+23+84+24+2+59=226

SCAN

14 37 53 65 67 98 122 124 183

53 -> 65 -> 67 -> 98 -> 122 -> 124

-> 183 -> 199 -> 37 -> 14

65-53=12

67-65=2

98-67=31

122-98=24

124-122=2

183-124=59

199-183=16

199-37=162

37-14=23

14-0=14

12+2+31+24+2+59+16+162+23+14=345

שיטת C- SCAN

עובדת באופן הבא-

\*\*ראש קריאה מתחיל לנוע מנקודת המוצא מסילה מסילה לכיוון הצילינדר האחרון כאשר הראש הגיע לצילינדר האחרון, באופן אוטומטי הוא עושה חזרה לצילינדר 0. ומצילינדר 0 הראש מתחיל לנוע לכיוון נקודת המוצא כאשר בדרך אם הוא פוגש בקשות, הוא מבצע כתיבה מצילינדר של בקשות.\*\*

מתחילה מנקודת המוצא ועוברת עד הצילינדר האחרון(כמו SCAN).

ברגע שהגיעה לצילינדר האחרון הראש קריאה כתיבה קופצת לצילינדר 0.

ברגע שהגיעה לצילינדר 0 היא נעה בחזרה לקריאה כתיבה.

53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199 -> 0 -> 14 -> 37

65-53=12

67-65=2

98-65=31

122-98=24

124-122=2

183-124=59

199-183=16

199-0=199

14-0=14

37-14=23

12+2+31+24+2+59+16+199+14+23=382

ישנה שיטה נוספת שצריכה להיות חסכונית יותר משיטת SCAN ו C SCAN

C-LOOK

שיטה זו פועלת בדומה לשיטות SCAN . אבל המסגרת שנבנית עבורה, מוגדרת ע"י צילינדר הכי נמוך בבקשות עד לצילינדר הכי גבוה שבבקשות.

כלומר, המסגרת של C LOOK במקרה שלנו יהיה מ-14 עד 183.

14 37 53 65 67 98 122 124 183

חישוב של C- LOOK -

53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 14 -> 37

65-53=12

67-65=2

98-67=31

122-98=24

124-122=2

183-124=59

183-14=169

37-14=23

12+2+31+24+2+59+169+23=322

RAID- REDUNDENT ARRAY OF INDEPENDENT DISKS- מערך יתיר(נוסף) של דיסקים עצמאיים

לוקחים כמה דיסקים עצמאיים ועובדים איתם. נועדו לשפר ביצועים של המערכת.

RAID 0 – (חיסכון בזמן)

במקום דיסק אחד משתמשים בכמה והעבודה נעשית באופן הבא:

מתחילים לעבוד עם דיסק 1 , ממלאים את הצילינדר הראשון בדיסק אחד. עצרנו, עוברים לדיסק 2 ממלאים \ קוראים ועוברים לדיסק 3. קראנו 0 0 0 בכל הדיסקים.

לאחר מכן נעשית הזזת זרוע לצילינדר הבא. חוסכים זמן של הזזת זרוע בין דיסקים.

לאחר מכן נעשית כתיבה של דיסק 1 2 ו 3 וכו'.

בעצם חוסכים 2 הזזות זרוע בין הדיסקים . מזיזים רק זרוע אחת כל 3 דיסקים.

RAID 1- mirror דיסק מראה- (גיבוי ואמינות)

יש לפחות 2 דיסקים ונעשית קריאה או כתיבה. מה שכותבים בדיסק הראשון כותבים גם בשני. (גיבוי)

כאשר אחד הדיסקים נדפק, יש לנו גיבוי ואז אפשר להמשיך לכתוב.

השיטה מאפשרת יצירת גיבוי אוטומטית.

RAID 3 –

אמינות המידע על הדיסקים

אנו זקוקים ל3 דיסקים

דיסק השלישי מכיל את הזוגיות

הראשון והשני ישנה מידע שאנחנו מבצעים עליו את חישוב הזוגיות ואת הזוגיות רושמים בדיסק השלישי

מידי פעם נעשית בדיקה בין התכנים של דיסק 1 ודיסק 2 מול דיסק 3- בצורה זו ניתן לוודא שהמידע תקין או לא תקין. במידע לא תקין בעזרת PARITY ניתן לשחזר את התכנים האמיתיים בדיסקים 1 ו2.

שחזור תוכן של מידע שהשתבש במהלך העבודה.

דוגמא שמסבירה את המשמעות של PARITY

בתקשורת מתבצע משלוח של 8 ביט בצורה כזו-

Start bit, אחרי 8 ביטים של התוכן שולחים parity ואחריו stop. הכל נשלח על פני קו אחד. (11 ביטים)

מה זה PARITY

אנחנו סופרים כמה 1 יש במשלוח. אם כמות ה-1 מספר זוגי PARITY ביט יהיה 1

אם כמות ה-1 אי זוגי PAIRTY ביט יהיה 0

כאשר ביטים נעים על פני הקו ישנה סבירות שאחד הביטים ישתבש. שני ביטים לא יכולים להשתבש.

בעקבות השיבוש, ה PARITY משתנה.

אם בדיסק הראשון או השני יש אין התאמה, אחד הביטים השתבש. אך יש אפשרות לשחזר את זה.

RAID 3 היא שיטה שאנחנו כותבים מידע על הדיסק הראשון והשני, עושים פעולה על נתונים בדיסקים, מחשבים את ה PARITY ושומרים את ה PARITY על דיסק שלישי.

RAID 5- (אמינות)

דורש לפחות 3 דיסקים ומבוסס על חישוב ה PARITY (זוגיות)

הזוגיות נשמרת בדיסק 3 ומידי פעם מתבצע חישוב הזוגיות של מידע שנמצא על דיסק 1 ו-2.

שיטת חישוב הזוגיות שונה ויותר מסובכת מRAID 3 ונותנת תוצאות שיחזור יותר טובות.

כל שיטות RAID נועדו לשפר את הביצועים של המערכת

שיטות איבוד של קריאת מידע בין המסילות על מנת לחסוך בתנועות הזרוע

FCFS

SSFT

SCAN

תרגיל-

מספרים ממוספרים מ0 עד 4999 (דיסק שמכיל 5000 צילינדר)

הבקשות לקריאת תכנים מהדיסק הן הבאות-

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

FCFS, SSFT, SCAN

כאשר הזרוע נמצאת בצילינדר 2150 ולפניו יש 1850. (כלפי מעלה)

FCFS-

2150 -> 2069 -> 1212 -> 2296 -> 2800 -> 544 -> 1618 -> 356 -> 1523 -> 4965 -> 3681

2150 – 2069=81

2069 – 1212=857

2296 – 1212=1084

2800 – 2296=504

2800 – 544=2256

1618 – 544=1074

1618-356=1262

1523-356=1167

4965-1523=3442

4965-3681=1284

81+857+1084+504+2256+1074+1262+1167+3442+1284=13,011

SSFT

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

2150 -> 2069 -> 2296 -> 2800 -> 3681 -> 4965 -> 1618 -> 1523 -> 1212 -> 544 -> 356

2150 – 2069=81

2296 – 2069=227

2800-2296=504

3681-2800=881

4965-3681=1284

4965-1618=3347

1618-1523=95

1523-1212=311

1212-544=668

544-356=188

7586

SCAN

2150 – ראש קריאה כתיבה

1805- תנועה למעלה

1805 -> 2150

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965,3681

356 -> 544 -> 1212 -> 1523 -> 1618 -> 2069 -> 2150 -> 2296 -> 2800 -> 3681 -> 4965

2150 -> 2296 -> 2800 -> 3681 -> 4965 > 4999-> 2069 -> 1618 -> 1523 -> 1212 -> 544 -> 356

2296-2150=146

2800-2296=504

3681-2800=881

4965-3681=1284

4999-4965=34

4999-2069=2930

2069-1618=451

1618-1523=95

1523-1212=311

1212-544=668

544-356=188

7474

C-SCAN

2150 – ראש קריאה כתיבה

1805- תנועה למעלה

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

2150 -> 2296 -> 2800 -> 3681 -> 4965 > 4999-> 0 -> 356 -> 544 -> 1212 -> 1523 -> 1618 -> 2069

2296-2150=146

2800-2296=504

3681-2800=881

4965-3681=1284

4999-4965=34

4999-0=4999

356-0=356

544-356=188

1212-544=668

1523-1212=311

1618-1523=95

2069-1618=451

9917

C-look

2150 – ראש קריאה כתיבה

1805- תנועה למעלה

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

2150 -> 2296 -> 2800 -> 3681 -> 4965 > 356 -> 544 -> 1212 -> 1523 -> 1618 -> 2069

2296-2150=146

2800-2296=504

3681-2800=881

4965-3681=1284

4965-356=4906

544-356=188

1212-544=668

1523-1212=311

1618-1523=95

2069-1618=451

9434

מטלה:

עבור אותה סידרה של בקשות , להתחיל תנועה מ2150 כלפי מטה עבור 3 שיטות

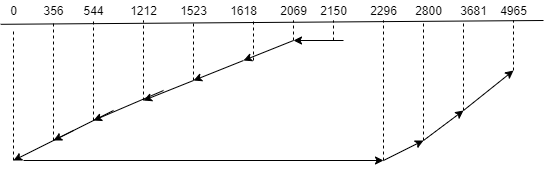
SCAN, C-SCAN, C-LOOK

**SCAN**

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

356 -> 544 -> 1212 -> 1523 -> 1618 -> 2069 -> 2150 -> 2296 -> 2800 -> 3681 -> 4965

2150 -> 2069 -> 1618 -> 1523 -> 1212 ->544 -> 356 -> 0 -> 2296 ->2800 ->3681 ->4965



חישוב-

2150-2069=81 .1

2069-1618=546 .2

1618-1523=95 .3

1523-1212=311 .4

1212-544=668 .5

544-356=188 .6

356-0=356 .7

2296-0=2296 .8

2800-2296=504 .9

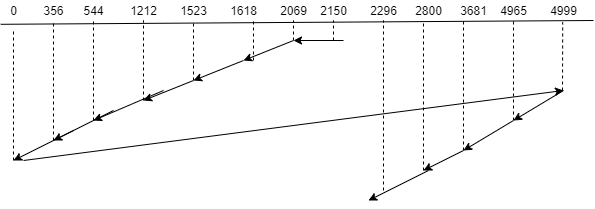
3681-2800=881 .10

4965-3681=1284 .11

**7210**

**C-SCAN**

2150 -> 2069 -> 1618 -> 1523 -> 1212 ->544 -> 356 -> 0 ->4999->4965->3681->2800->2296



חישוב-

2150-2069=81 .1

2069-1618=546 .2

1618-1523=95 .3

1523-1212=311 .4

1212-544=668 .5

544-356=188 .6

356-0=356 .7

4999-0=4999 .8

4999-4965=34 .9

4965-3681=1284 .10

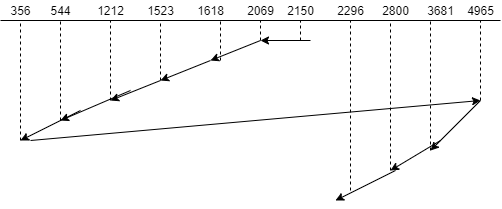
3681-2800=881 .11

2800-2296=504 .12

**9947**

**C-Look**

2150 -> 2069 -> 1618 -> 1523 -> 1212 ->544 -> 356 ->4965->3681->2800->2296



**חישוב-**

2150-2069=81 .1

2069-1618=546 .2

1618-1523=95 .3

1523-1212=311 .4

1212-544=668 .5

544-356=188 .6

4965-356=4609 .7

4965-3681=1284 .8

3681-2800=881 .9

2800-2296=504 .10

**9167**

מטלה

נתון דיסק עם 4000 צילינדרים מ0 עד 3999

ראש קריאה כתיבה של הדיסק נמצא בצילינדר 2000

הבקשה האחרונה שטופלה נמצאת בצילינדר 1900

הווקטור כלפי מעלה.

שיטות:

FCFS, SSFT, SCAN, C-SCAN, C-LOOK

רשימת הבקשות

2100, 2500, 1000, 800, 3000, 1400, 3100, 3200

**FCFS-**

800 -> 1000 -> 1400 -> 2000 -> 2100 -> 2500 -> 3000 -> 3100 -> 3200

2100-2000=100

2500-2100=400

2500-1000=1500

1000-800=200

3000-800=2200

3000-1400=1600

3100-1400=1700

3200-3100=100

7800

**SSFT-**

2100, 2500, 1000, 800, 3000, 1400, 3100, 3200

800 -> 1000 -> 1400 -> 2000 -> 2100 -> 2500 -> 3000 -> 3100 -> 3200

2000 -> 2100 -> 2500 -> 3000 -> 3100 -> 3200 -> 2000 -> 1400 -> 1000 -> 800

2100-2000=100

2500-2100=400

3000-2500=500

3100-3000=100

3200-3100=100

3200-2000=1200

2000-1400=600

1400-1000=400

1000-800=200

3600

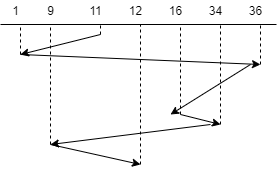
40 צילינדרים

ראש קריאה כתיבה 11

וקטור כלפי מעלה

1, 36, 16, 34, 9, 12

1, 9, 11, 12, 16, 34, 36



FSFC

11-1=10

36-1=35

36-16=20

34-16=18

34-9=25

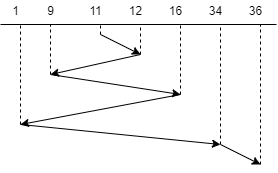
12-9=3

111

FFST

1, 9, 11, 12, 16, 34, 36

11,12,9,16,1,34,36



12-11=1

12-9=3

16-9=7

16-1=15

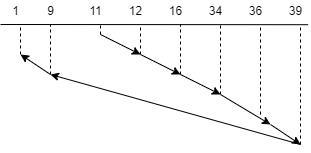
34-1=33

36-34=2

61

SCAN

1, 9, 11, 12, 16, 34, 36,39



12-11=1

16-12=4

34-16=18

36-34=2

39-36=3

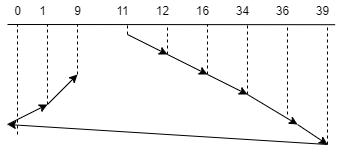
39-9=30

9-1=8

66

C-SCAN

0,1, 9, 11, 12, 16, 34, 36,39



12-11=1

16-12=4

34-16=18

36-34=2

39-36=3

39-0=39

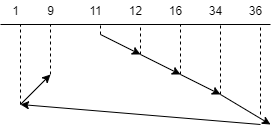
1-0=1

9-1=8

76

C-LOOK

1, 9, 11, 12, 16, 34, 36,



12-11=1

16-12=4

34-16=18

36-34=2

36-1=35

9-1=8

68

מידע ניהולי בטבלת הדפים

V-VALID ID-

1. לא נמצא
2. נמצא

R-REFERENCE BIT

0-לא היתה גישה

1-היתה גישה

M- MODIFY BIT

0-לא היה שינוי

1-היה שינוי

P- PROTECTION BIT

r- מותרת רק קריאה

Read/write- מותרת רק קריאה כתיבה

0-לא מוגן לכתיבה

1-דף מוגן לכתיבה

EXECUTE- קובץ ביצועים – מותר לבצע אותו

ישנה מערכת בעלת 8 דפים וברגע הנוכחי, התוכן של PAGE TABLE נראה כך

**-PAGE FRAME** – כתובת הפיזית הראשית ששם נמצא VIRTUAL PAGE

VIRTUAL PAGE VALID PAGE FRAME PROTECTION MODIFY BIT

0 1 #40 1 0

1 0

2 0

הפך מ-0 ל-1 בהמשך- 3 1 #200 1 0

4 1 #60 1 0

5 1 #8 EXECUTE 1

6 1 #70 0 0

7 1 #72 1 0

טבלת TLB

על מנת ליעל את המנגנון החלפת הדפים בנוסף ל PAGE TABLE, ישנה טבלת TLB

TLB יושבת בזיכרון המטמון (CASH MEMORY)

ה- CASH MEMORY בעל תכונה שהוא זיכרון מהיר- פי 10

לכן, החיפוש של הדף בתוך הזיכרון עובר מסלול קודם בטבלת TLB, אם הוא לא נמצא בTLB עוברים לטבלת PAGE TABLE

אם הדף נמצא ב-TLB, חסכנו המון זמן ושולפים משם את הכתובת של הזיכרון הראשי

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VALID** | **VIRTUAL PAGE** | **MODIFIED BIT** | **PROTECTION** | **PAGE FRAME** |
| 1 | 7 | 1 | 1 | 72 |
| 1 | 4 | 0 | 1 | 61 |
| 0 | 0 | 0 | 1 | 40 |
| 0 | 6 | 0 | 0 | 70 |

אם במהלך העבודה נצטרך להכניס דפים חדשים לתוך הזיכרון, הכתובות 200, 201, 202 יהיו מוכנסים ע"י הדפים הקיימים

כל דף חדש שהוא נכנס לתוך הזיכון, נעשה לו עדכון גם ב-TLB כאשר בתוך ה-TLB נתחיל לפנות את הדפים מתחילת הטבלה.

1. מוסיפים את VISUAL PAGE 3 – הוא החליף את 7 (הראשון ברשימה)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VALID** | **VIRTUAL PAGE** | **MODIFIED BIT** | **PROTECTION** | **PAGE FRAME** |
| 1 | 3 | 0 | 0 | 200 |
| 1 | 4 | 0 | 1 | 61 |
| 0 | 0 | 0 | 1 | 40 |
| 0 | 6 | 0 | 0 | 70 |

1. כתיבה לדף 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VALID** | **VIRTUAL PAGE** | **MODIFIED BIT** | **PROTECTION** | **PAGE FRAME** |
| 1 | 3 | 0 | 0 | 200 |
| 1 | 4 | 0 | 1 | 61 |
| 1 | 0 | 1 | 1 | 40 |
| 0 | 6 | 0 | 0 | 70 |

5. קריאת תוכן מדף 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **VALID** | **VIRTUAL PAGE** | **MODIFIED BIT** | **PROTECTION** | **PAGE FRAME** |
| 1 | 3 | 0 | 0 | 200 |
| 1 | 5 | 1 | EXECUTE | 8 |
| 1 | 0 | 1 | 1 | 40 |
| 0 | 6 | 0 | 0 | 70 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FRAME | TIME | V | R | M |
| 0 | 6 | 1 | 0 | 1 |
| 1 | 9 | 1 | 1 | 0 |
| 2 | 9 | 1 | 1 | 1 |
| 3 | 7 | 1 | 0 | 0 |
| 4 | 4 | 0 | 0 | 0 |

הכנסה של FRAME 4 במקום 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FRAME | TIME | V | R | M |
| 0 | 10 | 0 | 0 | 1 |
| 1 | 9 | 1 | 1 | 0 |
| 2 | 9 | 1 | 1 | 1 |
| 3 | 7 | 1 | 0 | 0 |
| 4 | 10 | 1 | 0 | 0 |

**שאלה**

מחשב מקצה לכל תהליך 65536בית(64kbyte)

כל הזיכרון מחולק לדפים של 4096 בית כל דף

לתוכנית מסוימת ישנו טקסט – 32768 בית

ישנם נתונים-16386 בית

ישנו סטק- 15870 בית

כל חלק צריך להיות בדפים נפרדים (חלק זיכרון משלו)

האם התוכנית הזאת יכולה להיכלל הזיכרון כאשר כל דף 4096

חישוב-

65536 / 4096 =16

יש סה"כ 16 דפים

32768 / 4096 = 8

16386 / 4096 = 4

15870 / 4096 = 4

8+4+4=16

הכל נכנס

מטריצה

D[64][64]

00,01,02…0 63

10,11,12…1 63

20,21,22…2 63

…

63 0, … 63 63

שורות-

כל דף 4kbyte - 4096

כל פריט 2 בייט

2\*64=128 byte - 128 גודל כל שורה

4096 / 128 =32 32 זה PAGE אחד

64\32 = 2

יש 2 החלפות דפים

עמודות-

2\*64=128

יש 128החלפות דפים