

ניתן להסיק שהעלות של קריאה בגישה אקראית היא הגבוהה ביותר בהקצאה משורשרת, כיוון שגם בקריאה מאמצע הקובץ, וגם מסופו, יש לעבור על פני חלק גדול מהקובץ כדי להגיע לנקודה המבוקשת. לגבי כתיבה אקראית, העלות היא הגבוהה ביותר בהקצאה רציפה, כיוון שגם בכתיבה לאמצע הקובץ וגם לתחילתו, יש להזיז בלוקים רבים, ולפעמים להזיז את הקובץ כולו. בסיכום כולל נראה שעלות של כתיבה/קריאה אקראית היא הגבוהה ביותר בהקצאה משורשרת, כי בין בקריאה ובין בכתיבה יש לעבור על כל הקובץ עד הנקודה הרצויה.

3.

יש לבדוק עבור כל קובץ מהן רוב הפעולות קלט/פלט שנעשות עליו, ולאיזה חלק של הקובץ ניגשים רוב הפעמים. זאת, מכיון שבכל שיטה יש יתרון כאשר מבצעים פעולה על חלק מסוים של הקובץ, וחיסרון, כאשר מבצעים פעולה אחרת על חלק אחר של הקובץ. דוגמאות:

(א) קובץ ריצה המכיל פקודה של מערכת ההפעלה, כמו cp, rm, chmod ב Unix. הקובץ אינו משתנה (כל זמן שזו אותה מערכת הפעלה, שזה זמן ארוך מאוד). לכן אין חשיבות לכתיבה על קובץ כזה. הקריאה תיעשה תמיד מתחילת הקובץ עד סופו, כי זהו קובץ קצר שצריך להעלות את כולו לזיכרון, לכן עדיף להשתמש בהקצאה רציפה- שהיא הפשוטה ביותר ומאפשרת גישה מהירה לתחילת הקובץ, שזה מה שצריך בקובץ כזה.

(ב) קובץ המכיל מאגר נתונים, כמו קובץ של גליון אלקטרוני. הקובץ משתנה באופן תדיר, והשינויים נעשים בכל מקום בקובץ. לפעמים חשוב לשלוח נתון דווקא מסוף הקובץ. לקובץ כזה ברור שעדיף להשתמש בהקצאת אינדקס, שתאפשר גם קריאה מהירה מסוף הקובץ, וגם כתיבה מהירה באמצע או תחילת הקובץ.

(ג) קובץ ריצה של תוכנית כלשהי כמו word, explorer. זהו קובץ שלא משתנה, והקריאה נעשית מתחילתו ועד סופו. בד"כ זהו קובץ גדול, לכן עדיף להשתמש בהקצאה משורשרת, כי בהקצאה רציפה לא בטוח תמיד שיש מקום רצוף ומספיק ארוך. אין צורך לקפוץ לאמצע הקובץ או לסופו, אלא תמיד קוראים אותו לפי הסדר, לכן אין יתרון בהקצאת אינדקס, והגישה כל פעם לבלוק האינדקס כדי לדעת מה הבלוק הבא מבזבזת זמן.

(ד) קובץ ששומר הגדרות כמו config.sys, autoexec.bat ב Dos. זהו קובץ שמתבצעות אליו גם קריאה וגם כתיבה בכל מקום בקובץ. לכן ברור שעדיף להשתמש בהקצאת אינדקס.

(ה) קובצי cookies ב-windows. אלו קבצים קטנים שנכתבים ונקראים בשלמותם באופן תדיר מאוד. לכן עדיף להשתמש בהקצאה רציפה ע"מ להמנע מ-over-head של התעסקות עם מצביעים. לא צריכה להתעורר בעיה של מקום כי הקובץ קטן מאוד.

(ו) קובצי תמונות. אלו קבצים גדולים מאוד שנקראים בשלמותם ואין משמעות לקריאה ממקום שאינו תחילת הקובץ. כמעט ואין כתיבה של חלקים קטנים אלא או הכל או כלום (ובד"כ אין כתיבה כלל). לכן עדיף להשתמש בהקצאה משורשרת, שתדע לנהל נכון מספיק מקום בשביל הקובץ, ותאפשר גישה רציפה מהירה כדי לקרוא את כל הקובץ בשלמותו.

(ז) קבצי וידאו/אודיו. אלו קבצים ענקיים שיש צורך לגשת לפעמים לאמצע קובץ. השימוש בקובץ נעשה לקריאה וכמעט ולא מתבצעת כתיבה. לכן עדיף להשתמש בהקצאת אינדקס שתנצל את המקום בדיסק ותתן אפשרות לגישה מהירה לכל חלקי הקובץ.

4.

(א) תתבצענה 4 קריאות:

- I- קריאת קובץ הספרייה של השורש המוצבע תמיד ע"י I-node אפס. בקובץ של השורש, נמצא את מספר ה-I-node המתאים לספרייה etc. ניגש ל I-node זה, שמצביע לקובץ הספרייה.
- II- קריאת קובץ הספרייה etc ע"מ למצוא את מס' ה-I-node של הקובץ passwd. ניגש ל I-node זה שמצביע לקובץ עצמו.
- III- קריאת ה-I-node של הקובץ passwd.
- IV- קריאת הקובץ עצמו (ה-data block הראשון).

(ב) בניגוד למקרה הקודם, כאן אנחנו נמצאים בתוך ספריית usr, לכן תתבצענה 3 קריאות:

- I- קריאת קובץ הספרייה של usr, "..", כדי למצוא את מספר ה-I-node של הקובץ file. ניגש ל I-node זה שמצביע לקובץ עצמו.
- II- קריאת ה-I-node של הקובץ עצמו.
- III- קריאת הקובץ עצמו.

(ג) תתבצענה 4 קריאות:

- I- קריאת קובץ הספרייה של השורש המוצבע תמיד ע"י I-node אפס. בקובץ של השורש, נמצא את מספר ה-I-node המתאים לספרייה tmp. ניגש ל I-node זה, שמצביע לקובץ הספרייה.
- II- קריאת קובץ הספרייה tmp ע"מ למצוא את מס' ה-I-node של הקובץ foo. ניגש ל I-node זה שמצביע לקובץ עצמו.
- III- קריאת ה-I-node של הקובץ tmp.
- IV- קריאת הקובץ עצמו.

5.

(א) גודל הקובץ המקסימלי הוא: גודל כל בלוק * מספר הבלוקים. מספר הבלוקים הוא 10 (כי כל כניסה מצביעה ישירות לבלוק אחד), לכן גודל הקובץ המקסימלי הוא: $10 * 4,096 = 40,960$ בתים. ה-I-node עצמו בגודל $10 * 4 = 40$ בתים. לכן בסה"כ הקובץ תופס 41,000 בתים.

(ב) גודל ה-indirect block הוא 1,024 בתים. כל כניסה בגודל 4 בתים. לכן יש לו $1,024 / 4 = 256$ כניסות. ז"א מספר הבלוקים הוא: $66,055 = 1 * 256^2 + 2 * 256 + 7$. לכן גודל הקובץ המקסימלי הוא: $66,055 * 1,024 = 67,640,320$ בתים. ה-

I-node עצמו תופס $10 \cdot 4 = 40$ בתים, וה- indirect blocks תופסים $1,024 \cdot 3 + 1,024 \cdot 256 = 265,216$ בתים, לכן סה"כ הקובץ תופס 67,905,576 בתים.

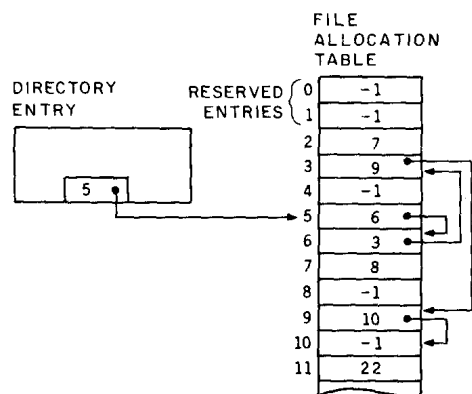
6.

הוספה ב-	הקצאה רציפה	הקצאה משורשרת	אינדקס
תחילת הקובץ	(א) קריאת 100 בלוקים. (ב) כתיבת הבלוק החדש. (ג) כתיבת 100 הבלוקים לאחריו. סה"כ: 201 פעולות	(א) קריאת טבלת הפנויים ע"מ להקצות מקום לכתיבה. (ב) כתיבת הבלוק החדש. (ג) קריאת המצביע ב fd. (ד) כתיבת מצביע זה בסוף החלק החדש. (ה) כתיבת כתובת התא החדש למצביע של ה fd. סה"כ: 1 פעולות	(א) קריאת טבלת הפנויים ע"מ להקצות מקום לכתיבה. (ב) כתיבת הבלוק החדש. (ג) עדכון ה I-node של הקובץ. אם עשרת המקומות הראשונים כבר מלאים יש לקרוא indirect block (או להקצות אותו) וכן הלאה כמספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות
אמצע הקובץ	(א) קריאת 50 בלוקים. (ב) כתיבת הבלוק החדש. (ג) כתיבת 50 הבלוקים לאחריו. סה"כ: 101 פעולות	(א) קריאת טבלת הפנויים ע"מ להקצות מקום לכתיבה. (ב) כתיבת הבלוק החדש. (ג) קריאת 51 מצביעים (כדי להגיע למצביע למקום ה-51). (ד) כתיבת המצביע לבלוק ה-51 בסוף הבלוק החדש. (ה) שינוי המצביע בבלוק הקודם כך שיצביע לבלוק החדש. סה"כ: 2 פעולות	(א) קריאת טבלת הפנויים ע"מ להקצות מקום לכתיבה. (ב) כתיבת הבלוק החדש. (ג) עדכון ה I-node של הקובץ. אם עשרת המקומות הראשונים כבר מלאים יש לקרוא indirect block (או להקצות אותו) וכן הלאה כמספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות
סוף הקובץ	פעולה אחת- כתיבת החלק החדש	(א) קריאת טבלת הפנויים ע"מ להקצות מקום לכתיבה. (ב) כתיבת הבלוק החדש. (ג) קריאת הבלוק האחרון (ההנחה היא שיש גישה ישירה לסוף הקובץ). (ד) שינוי המצביע בבלוק האחרון (לשעבר) כך שיצביע לבלוק החדש. סה"כ: 2 פעולות	(א) קריאת טבלת הפנויים ע"מ להקצות מקום לכתיבה. (ב) כתיבת הבלוק החדש. (ג) עדכון ה I-node של הקובץ. אם עשרת המקומות הראשונים כבר מלאים יש לקרוא indirect block (או להקצות אותו) וכן הלאה כמספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות

7.

FAT הם ר"ת של File Allocation Table. בשיטה זה מערכת ההפעלה מחלקת כל דיסק לאשכולות- Clusters. זוהי חלוקה לוגית בה כל אשכול מכיל כמות מסוימת של בלוקים של הדיסק- Sectors. בתחילת הדיסק מאוחסנת טבלת ה- FAT. בגלל חשיבותה מאוחסן תמיד עוד עותק שלה בצמוד לטבלה המקורית. לאחר מכן מאוכסן חלק ה- Directory, שם לכל קובץ יש כניסה- Entry. בכניסה מופיע שם הקובץ, גודלו והאשכול הראשון ששייך לקובץ. כדי לדעת מהו האשכול הבא יש לפנות לטבלת ה- FAT. הטבלה היא בעצם מערך חד-מימדי בו כל תא מייצג אשכול בדיסק. כל תא מכיל את מספר האשכול הבא של הקובץ שאליה יש לגשת. כדי לסמן סוף קובץ, התא של האשכול האחרון מכיל 1-0. אשכול שאינו מנוצל ע"י שום קובץ מכיל 0.

לדוגמא, נניח שהאשכול הראשון שמוצבע ע"י הכניסה בספרייה הוא 5. לאחר מכן הקובץ תופס את אשכול 6, 3, 9 ו-10. האיור הבא מדגים כיצד זה יראה ב- FAT:



אם הקובץ הוא קובץ ספרייה, הוא פשוט יכיל מצביע למקום אחר בדיסק שם ישנה Directory Table של תת-הספרייה, שתכיל את המצביעים לקבצים שבספרייה זו.

שיטה זו היא בעצם ייעול שיטת ההקצאה המשורשרת. במקום שכל בלוק יכיל מצביע לבלוק הבא, וכך נאלץ לעבור על כולם כדי להגיע לסוף הקובץ, ה- FAT שומר את הסדר, ונצטרך רק לעבור עליו. כל כניסה ב- FAT תופסת מעט מאוד מקום- 16 או 32 Bit, כך שניתן להעלות את כל ה- FAT לזיכרון, ואז אין צורך לגשת כל פעם לדיסק כדי למצוא מיקום מסוים בתוך הקובץ. בתוך כל אשכול, שמכיל כמה בלוקים, ניתן לגשת אל הבלוקים בגישה ישירה. עדיין נשמר היתרון של הקצאה משורשרת- קל להוסיף מידע באמצע או תחילת הקובץ.