

תרגיל תיאורטי 1

1. טבלת היתרונות:

אינדקס	הказאה משורשת	הказאה רציפה	
(א) מכיוון שאין צורך לשמר על רציפות, ניתן להרכיב את הקובץ מקטעים קטנים וכך לנצל "חללים" ריקים בדיסק ולמלא אותם. ריקם בדיסק ולמלא אותם. (ב) ל-node-I גודל קטן וקבוע.	(א) מכיוון שאין צורך לשמר על רציפות, ניתן להרכיב את הקובץ מקטעים קטנים וכך לנצל "חללים" ריקים בדיסק ולמלא אותם. (ב) ניתן, באופן תיאורטי, ליצור קבצים ענקיים ככל שנרצה (אם המקום בדיסק מאפשר).	(א) הדיסק מסודר יותר, מכיוון שכל קובץ נמצא במקומ אחד בלבד והוא לא מחולק לקטעים קטנים המפוזרים בכמה מקומות. יש פחת צורך לבצע פעולה איחוי (defrag). (ב) לכלי קובץ נשמר אך ורק מצביע אחד, לכן רוב המיקום בדיסק יכול להיות מנוצל לבתונים עצם.	מקום בדיסק
(א) גישה מהירה לכל מקום בקובץ, ללא מעבר מיותר על הנתונים שלא מחפשים. (ב) ניתן גם לגשת לנתונים באופן סידרתי. (ג) קל לבצע שינויים בכל מקום בקובץ.	(א) ציריך לקרוא רק מצביע אחד ע"מ לדעת את מיקומו של הקובץ. (ב) ניתן לגשת לנ נתונים ברציפות - גישה סדרתית - באופן יעיל. (ג) קל לבצע שינויים בכל מקום בקובץ.	(א) ציריך לקרוא רק מצביע אחד ע"מ לדעת את מיקומו של הקובץ בלבד. (ב) גישה מהירה מאוד לכל מקום בקובץ - גישה ישירה - ללא מעבר מיותר על נתונים שלא מחפשים.	גישה יעילה

2.

השיטה	בתחילת קובץ		באמצע קובץ		בסוף קובץ		השיטה
	כתובת	קריאה	כתובת	קריאה	כתובת	קריאה	
רציפה	גישה מהירה מאוד, כי ניגשים ישר לכטבות הקובץ באופן ישיר- כתובות ההתחלה + חצי מאורך הקובץ.		גישה מהירה מאוד, כי ניגשים ישר לכטבות הקובץ באופן ישיר- כתובות ההתחלה + חצי מאורך הקובץ.		גישה מהירה מאוד, כי ניגשים ישר לכטבות הקובץ.		הказאה רציפה
	אם המיקום בהמשך תפוס לא נכל להמשר לכתוב ונאלץ להעתיק את כל הקובץ למקום אחר.		צריך להזיז חצי מהקובץ ע"מ כתוב. כמו"כ אם המיקום בהמשך תפוס, נאלץ להעתיק את כל הקובץ למקום אחר.		צריך להזיז ע"מ כל הקובץ כתוב.		
הказאה משורשת	פשוט מוסיפים בלוק ומעודכנים את המצביעים בלי לגעת בשאר הקובץ.		פשוט מוסיפים בלוק ומעודכנים את המצביעים בלי לגעת בשאר הקובץ.	כשנמצאים באיבר מסוים, יש גישה ישירה בשאר הקובץ.	פשוט מוסיפים בלוק ומעודכנים את המצביעים בלי לגעת בשאר הקובץ.	גישה מהירה מאוד, כי ניגשים ישיר למצביע הראשו.	הказאה משורשת
	גישה איטית- צריך לעبور באופן סידרתי על כל המצביעים.		גישה איטית- צריך לעبور באופן סידרתי על המצביעים.			גישה מהירה מאוד, כי ניגשים ישיר למצביע הראשו.	
אינדקס	פשוט מוסיפים בלוק ומעודכנים את המצביעים בלי לגעת בשאר הקובץ.	גישה מהירה, כי ניגשים לבlok האחרון ברשימה.	פשוט מוסיפים בלוק ומעודכנים את המצביעים בלי לגעת בשאר הקובץ.	גישה מהירה, כי ניגשים לבlok האמצעי ברשימה.	פשוט מוסיפים בלוק ומעודכנים את המצביעים בלי לגעת בשאר הקובץ.	גישה מהירה מאוד, כי ניגשים ישיר לבlok הראשו.	הказאה אינדקס
	צריך "לחפור" בעומק בטבלאות "ע"מ להציג מקום שמיירת המצביע. המצביע.	בקבצים גדולים גוררת הזזה של כל המצביעים הבאים אחרי למיקום הבא בטבלה שב-node-I. שוב החל מהצביעים משנה מיקום ולפעמים יש צורך בהקצתה indirect .blocks	בקבצים גדולים יש צורך לעبور על מסלול של מצביעים.	הוסף המצביע גוררת הזזה של כל מצביעים המצביעים למיקום הבא בטבלה שב-node-I. במרקחה שהטבלה 10 מיליה, חוץ מצביעים, יש צורך להקצות indirect .block			

ניתן להסיק שהעלות של קרייה בגישה אקראית היא הגבואה ביותר בהקצתה מושרשת, כיוון שגם בקריאה ממוצע הקובץ, וגם מסווגו, יש לעבור על פני חלק גדול מהקובץ כדי להגיע לנקודת המבוקשת. לבני כתיבה אקראית, העלות היא הגבואה ביותר בהקצתה רציפה, כיוון שגם בכתיבה לאמצע הקובץ גם לתחילת כתיבת הבלוקים רבים, ולפעמים להציג את הקובץ כולו. בסיכום כולל נראה שעלות של כתיבה/קרייה אקראית היא הגבואה ביותר בהקצתה מושרשת, כי בין לקריאה ובין בכתיבה יש לעبور על כל הקובץ עד הנקודה הרציפה.

.3

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

- (א) קובץ ריצה המכיל פקודת של מערכת הפעלה, כמו rm, chmod, cp, ב Unix. הקובץ אינו משתנה (כל זמן שהוא) למערכת הפעלה, שזה זמן אורך מאוד. לכן אין חשיבות לכתיבה על קובץ זהה. הקריאה תעשה תמיד מתחילת הקובץ עד סוףו, כי קובץ קצר לשכיר להעלות את כלו לyncron, لكن עדיף להשתמש בהקצתה רציפה - שהיא הפשטה ביותר ומאפשרת גישה מהירה לתחילת הקובץ, שזה מה שऋיך בקובץ זהה.
- (ב) קובץ המכיל מאגר נתונים, כמו קובץ של גלוןALKTRAN. הקובץ משתנה באופן תדיר, והשינויים נעשים בכל מקום בקובץ. לעיתים חשוב לשולב בתוך דואק מסוף הקובץ. לקובץ זהה ברור שעדיף להשתמש בהקצת אינדקס, שתאפשר גם קריאה מהירה מסוף הקובץ, וגם כתיבה מהירה באמצעותם של הקובץ. בד"כ
- (ג) קובץ ריצה של תוכנית כלשהי כמו word, explorer וכו'. קובץ שלא משתנה, והקריאה נעשית תמיד מנקודתו ועד סוףו. זה קובץ גדול, שכן עדיף להשתמש בהקצתה מושרשת, כי בהקצתה רציפה לא בטוח תמיד שיש מקום רצוף ומספיק מקום. אין צורך לкопיא לאמצע הקובץ או לסתופו, אלא תמיד קוראים אותו לפי הסדר, שכן אין יתרון בהקצת אינדקס, והגישה כל פעם בלבד האינדקס כדי לדעת מה הבלוק הבא מבחרת זמן.
- (ד) קובץ שומר הגדירות כמו config.sys, autoexec.bat ב-Dos. זה קובץ שמתבצעת אליו גם קריאה וגם כתיבה בכל מקום בקובץ. שכן ברור שעדיף להשתמש בהקצת אינדקס.
- (ה) קובצי cookies ב-windows. אלו קובצים קטנים שנכתבים ונקראים בשלהמאות באופן תדיר מאוד. שכן עדיף להשתמש בהקצתה רציפה ע"מ להמנע מ-over-head של התעוקות עם מצביעים. לא צריכה להתעורר בעיה של מקום כי הקובץ קטן מאוד.
- (ו) קובצי תמונות. אלו קובצים גדולים מאוד שנקראים בשלהמאות לקריאה ממוקם שאינו תחילת הקובץ. כמעט ואין כתיבה של חלקים קטנים אלא או הכל או כלום (ובד"כ אין כתיבה כל). שכן עדיף להשתמש בהקצתה מושרשת, שתடע לנחל נכון מקום בשביל הקובץ, וטאפר גישה רציפה כדי לקרוא את כל הקובץ בשלהמאות.
- (ז) קבצי וידאו/אודיו. אלו קובצים ענקים שיש צורך לגורש לפחות לפעמים לאמצע קובץ. השימוש בקובץ געה לקריאה וכמעט ולא מתבצעת כתיבה. שכן עדיף להשתמש בהקצת אינדקס שתנצל את המקום בדיסק ונתן אפשרות לגישה מהירה לכל חלקו של הקובץ.

.4

(א) תבצענה 4 קרייאות:

- I- קריית קובץ הספריה של השורש המוצבע תמיד ע"י node-I אפס. בקובץ של השורש, נמצא את מספר ה-node-I המתאים לספריה etc. ניגש/node-I זה, שמצויב לקובץ הספריה.
- II- קריית קובץ הספריה etc ע"מ למצוא את מס' ה-node-I של הקובץ passwd. ניגש/node-I זה שמצויב לקובץ עצמו.
- III- קריית node-I של הקובץ passwd.
- IV- קריית הקובץ עצמו (ה-block data最先).

(ב) בנויג לדמייה הקודם, כאן אנחנו נמצאים בתוך ספריית fs, אך ניתן תבצענה 3 קרייאות:

- I- קריית קובץ הספריה של usr, "...", כדי למצוא את מס' ה-node-I של הקובץ file. ניגש/node-I זה שמצויב לקובץ עצמו.
- II- קריית node-I של הקובץ עצמו.
- III- קריית הקובץ עצמו.

(ג) תבצענה 4 קרייאות:

- I- קריית קובץ הספריה של השורש המוצבע תמיד ע"י node-I אפס. בקובץ של השורש, נמצא את מספר ה-node-I המתאים לספריה tmp. ניגש/node-I זה, שמצויב לקובץ הספריה.
- II- קריית קובץ הספריה tmp ע"מ למצוא את מס' ה-node-I של הקובץ foo. ניגש/node-I זה שמצויב לקובץ עצמו.
- III- קריית node-I של הקובץ tmp.
- IV- קריית הקובץ עצמו.

.5

(א) גודל הקובץ המקורי הוא: גודל כל בלוק * מספר הבלוקים. מספר הבלוקים הוא 10 (כי כל כניסה מצביעה ישירות לבLOCK אחד), ולכן גודל הקובץ המקורי הוא: $40,960 = 40 \times 4,096 = 40 \times 10^4$ בתים. ה-node-I עצמו בגודל $40 = 4 \times 10$ בתים. לכן בסה"כ הקובץ TotSize הוא 41,000 בתים.

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

inode-I עצמו תופס $10^4 \times 4 = 1,024$ בתים, והוא indirect blocks תופס $1,024 \times 256 = 265,216$ בתים, לכן סה"כ הקובץ תופס $67,905,576$ בתים.

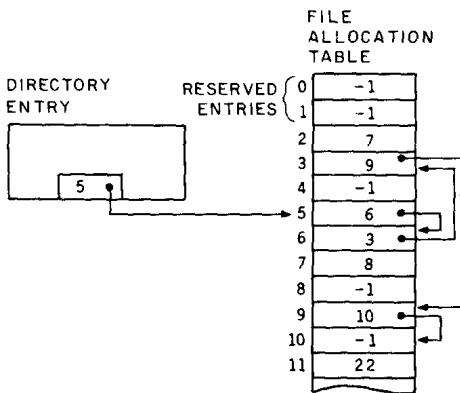
.6

הוספה ב-	הказאה רציפה	הказאה משורשת	אינדקס
תחילת הקובץ	סה"כ: 201 פעולות	(א) קריית טבלת הפנויים ע"מ להקצות מקום לכטיבה. (ב) כתיבת הבלוק החדש. (ג) עדכוןinode-I של הקובץ. אם عشرת המיקומות הראשונים כבר מלאים יש לקרוא indirect block (או indirect block) (או להקצות אותו) וכן הלאה במספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות	(א) קריית טבלת הפנויים ע"מ להקצות מקום לכטיבה. (ב) כתיבת הבלוק החדש. (ג) עדכוןinode-I של הקובץ. אם عشرת המיקומות הראשונים כבר מלאים יש לקרוא indirect block (או indirect block) (או להקצות אותו) וכן הלאה במספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות
אמצע הקובץ	סה"כ: 101 פעולות	(א) קריית 50 בלוקים. (ב) כתיבת הבלוק החדש. (ג) כתיבת 50 הבלוקים לאחריו. סה"כ: לפחות 1 פעולות	(א) קריית 50 בלוקים. (ב) כתיבת הבלוק החדש. (ג) עדכוןinode-I של הקובץ. אם عشرת המיקומות הראשונים כבר מלאים יש לקרוא indirect block (או indirect block) (או להקצות אותו) וכן הלאה במספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות
סוף הקובץ	פעולה אחת- כתיבת החלק החדש	(א) קריית טבלת הפנויים ע"מ להקצות מקום לכטיבה. (ב) כתיבת הבלוק החדש. (ג) קריית הבלוק האחרון (ההנחה היא שיש גישה לשירה לסופם של הקובץ). (ד) שניי המצביע בבלוק האחרון (לעתבר) כרך שיצבע לבlok החדש. סה"כ: 2 פעולות	(א) קריית טבלת הפנויים ע"מ להקצות מקום לכטיבה. (ב) כתיבת הבלוק החדש. (ג) עדכוןinode-I של הקובץ. אם عشرת המיקומות הראשונים כבר מלאים יש לקרוא indirect block (או indirect block) (או להקצות אותו) וכן הלאה במספר הבלוקים הדרושים. סה"כ: לפחות 1 פעולות

.7

FAT המ"ר של File Allocation Table. בשיטה זה המערכת הפעלה מחלקת כל דיסק לאשכולות- Clusters. זהוי חלוקה לוגית בה כל אשכול מכיל כמהות מסוימת של בלוקים של הדיסק- Sectors. בתחילת הדיסק מאוחסנת טבלת ה- FAT. FAT. בغالל חשיבותה מאוחסן תמיד עוד עותק שלה בזיכרון לטבלה המקורית. לאחר מכן מאוכסן חלק ה- Directory, שם לכל קובץ יש כניסה. בכניסה מופיע שם הקובץ, גודלו והאשכול הראשון ששיך 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 שלו. כדי לסייע בפניות מושך סוף קובץ, התא של האשכול האחרון מכיל 1. אשכול שאינו מנוצל ע"י שום קובץ מכיל 0.

לדוגמה, נניח שהאשכול הראשון שמוצבע ע"י הכניסה בספריה הוא 5. לאחר מכן הקובץ תופס את אשכול 6, 3, 9 ו- 10. האירור הבא מדגים כיצד זה נראה ב- FAT:



אם הקובץ הוא קובץ ספריה, הוא פשוט יוכל מצביע למקומות אחרים בדיסק שם ישנה Directory Table של תת-הספריה, שתכיל את המצביעים לפחות במספריה זו.

שיטת זו היא בעצם ייעול שיטת ההказאה המשורשת. במקרים של כל בלוק יוכל מצביע לבlok הבא, וכך נאלץ לעבור על כלום כדי להגיע לסופם של הקובץ, ה- FAT שומר את הסדר, ונוצר רק לעבור עלי. כל כניסה ב- FAT תופסת מעט מקום - 16 או 32 Bit. כדי שניתן להעלות את כל ה- FAT לחישוב, ואז אין צורך לגשת כל פעם לדיסק כדי למצוא מקום מסוים בתחום הקובץ. בטור כל אשכול, שמכיל כמה בלוקים, ניתן לגשת אל הבלוקים בגישה ישירה. עדין נשמר היתרון של ההказאה המשורשת - כל להוסף מידע במאוץ או תחילת הקובץ.