2 – שאלה 2

2.1 – יתרון של מדיניות LIFO

היתרון העיקרי של מדיניות תזמון זרוע הדיסק LIFO (Last In, First Out) הוא הפשטות שלה. זה יכול להוביל לזמני תגובה מהירים לבקשות שהוצאו לאחרונה, מה שעשוי להיות מועיל ביישומים מסוימים בזמן אמת או רגישים לזמן שבהם הנתונים העדכניים ביותר זקוקים לעיבוד מיידי. בנוסף, מדיניות LIFO יכולה להיות קלה יותר ליישום והבנה בהשוואה לאלגוריתמי תזמון מורכבים יותר.

2.2 – חסרון של מדיניות LIFO

החיסרון העיקרי של מדיניות LIFO הוא שהיא עלולה להוביל להרעבה של בקשות קודמות. מכיוון שהבקשה האחרונה תמיד מעובדת תחילה, בקשות ישנות יותר יכולות להמתין ללא הגבלת זמן אם בקשות חדשות ממשיכות להגיע. זה יכול לגרום להגדלה משמעותית של זמני המתנה לבקשות מסוימות, מה שיוביל לביצועי מערכת כלליים גרועים ולאי הוגנות. יתר על כן, LIFO יכול לגרום לתנועות לא יעילות של זרועות הדיסק, להגדיל את זמן החיפוש ולהפחית את התפוקה, שכן ייתכן שזרוע הדיסק תצטרך להפוך כל הזמן לכיוון כדי לטפל בבקשה האחרונה.

3 – שאלה 3

כדי להחליט איזו שיטת אחסון קבצים אופטימלית לפי קריטריונים שונים, ננתח כל שיטה בהתבסס על המאפיינים הנדרשים רציפה. שלוש השיטות שנסקור הן הקצאה רציפה, רשימה מקושרת ו-i-node.

תחילה נשים לב כי בהתחשב בכך שגודל הקובץ יכול להשתנות באופן דינמי, הקצאה רציפה אינה מעשית בשל רגישותה לריסוק. לכן, נתמקד בהשוואת הרשימה המקושרת ו-i-node.

א. בזבוז מקום מינימלי

1. רשימה מקושרת:
   - כל בלוק מכיל מצביע לבלוק הבא.
   - יתרון: אין ריסוק חיצוני; רק ריסוק פנימי בתוך הבלוק האחרון.
   - חסרון: נדרש מקום נוסף עבור מצביעים, מה שמפחית את קיבולת האחסון האפקטיבית.

2. i-node:
   - כל i-node מכיל רשימה של מצביעים לבלוקים של הקובץ.
   - יתרון: הקצאה גמישה יותר מאשר רציפה, הפחתת ריסוק חיצוני.
   - חסרון: חלק מהשטח משמש למצביעים בתוך מבנה ה-i-node.

חישוב לבזבוז מקום מינימלי:
- נניח טווח גודל קובץ של $4KB$ עד $4MB$.
- גודל בלוק טיפוסי: $4KB$.

לרשימה מקושרת:
- לכל בלוק של $4KB$ יש מצביע (למשל, 4 בתים, לשם הפשטות).
- עבור קובץ של $4MB$: $\frac{4MB}{4KB} = 1024$ בלוקים.
- תקורה של מצביע: $1024 * 4byte = 4KB$
- בזבוז: מינימלי.

עבור i-node:
- יכול לתת מענה למספר רב של בלוקים ישירות או דרך בלוקים עקיפים.
- ממזער ריסוק פנימי עם הקצאת בלוקים גמישה.
- התקורה עבור i-nodes פחות משמעותית בהשוואה לרשימות מקושרות עבור קבצים גדולים.

מסקנה לבזבוז מינימלי של מקום:
- i-node: מאזן בין יעילות שטח לגמישות, ומפחית ריסוק פנימי וחיצוני כאחד.

ב. זמנים גישות סדרתיות מינימליות

1. רשימה מקושרת:
- יתרון: יישום פשוט.
- חסרון: איטי יותר עקב מעבר מצביע עבור כל בלוק.

2. i-node:
- מצביעים ישירים ועקיפים מאפשרים גישה מהירה לבלוקים.
- יתרון: גישה רציפה מהירה יותר בזכות מצביעים ישירים.
- חסרון: תקורה קלה לגישה לבלוקים עקיפים, אך בדרך כלל יעילה.

חישוב זמן גישה:
לרשימה מקושרת:
- גישה לכל בלוק כוללת קריאת הבלוק ומעקב אחר המצביע.
- עבור קובץ של $4MB$ (1024 בלוקים), מעבר מצביע מוסיף תקורה.

עבור i-node:
- מצביעים ישירים מספקים גישה מהירה בדומה להקצאה רציפה.
- מצביעים עקיפים מוסיפים עיכוב מינימלי אך הם בדרך כלל יעילים.

סיכום:
- i-node: מספק גישה רציפה מהירה יותר עם תקורה מינימלית בהשוואה לרשימות מקושרות.

המלצה סופית
- לבזבוז מקום מינימלי: i-node.
- לזמני גישות מינימליות: i-node.
i-node עדיפה עבור שני הקריטריונים, ומציעה יעילות שטח טובה יותר וגישה רציפה מהירה יותר.

4 – שאלה 4

4.1 – דמיון ושוני

קווי דמיון:
- בידוד: שניהם מספקים בידוד בין יישומים/תהליכים.
- אבטחה: שניהם מכוונים לאבטח ולבודד תהליכים כדי למנוע הפרעות ולהגן על המערכת.

הבדלים:
- לקונטיינרים יכולות להיות מספר תהליכים מבודדים באמצעות מרחבי שמות PID, בעוד ש-Android מבודדת כל אפליקציה בנפרד.
- ניהול משאבים: קונטיינרים משתמשים בקבוצות cgroups ומרחבי שמות, בעוד אנדרואיד משתמש במנהל החבילות להרשאות ותקשורת בין תהליכים.

4.2 – סוג הבידוד

אנדרואיד מנצלת את ההגנה המבוססת על משתמש לינוקס כדי לזהות ולבודד את משאבי האפליקציה. זה מבודד את האפליקציות זו מזו ומגן על אפליקציות ועל המערכת מפני אפליקציות זדוניות. לשם כך, אנדרואיד מקצה מזהה משתמש ייחודי (UID) לכל אפליקציית אנדרואיד ומריצה אותה בתהליך משלה.

4.3 – היתרון של $Copy\ On\ Write\ (COW)$ במודל התהליך של אנדרואיד

COW מפחיתה את השימוש בזיכרון על ידי שיתוף דפים עד לשינוים, משפרת את הביצועים על ידי צמצום ההעתקה במהלך יצירת התהליך, ומייעלת את השימוש במשאבים.

5 – שאלה 5

5.1 – הליך לתרגום כתובת לוגית בעלת 32 סיביות לכתובת פיזית

שתי שיטות למניעת ניסיון שינוי ה capability הניתנת למשתמש במערכת ללא תמיכת חומרה הן רשימות יכולות המנוהלות על ידי מערכת הפעלה והצפנה.

א. רשימות יכולות המנוהלות על ידי מערכת הפעלה

תיאור:
ה-capability מאוחסנות ברשימה מוגנת המנוהלת על ידי מערכת ההפעלה. רשימות אלו נטענות לתוך שטח הזיכרון של מערכת ההפעלה, וניתן לבצע שינויים רק עם הרשאות ליבה.

יתרון:
- בקרה: המשתמש אינו יכול לבצע שינויים ישירות, ואין צורך בהצפנה, מה שחוסך זמן מחשוב.

חסרון:
- תלות: כל שינוי ביכולות חייב להתבצע על ידי מערכת ההפעלה, המחייב התערבות kernel לעדכונים.

ב. הצפנה

תיאור:
היכולות מוצפנות, מה שמבטיח שניתן לזהות ולמנוע כל שינוי לא מורשה ביכולות.

יתרון:
- אבטחה: מספק הגנה חזקה מפני שיבוש, שכן כל שינוי ביכולת יהפוך אותה לבלתי שמישה ללא הפענוח הנכון.

חסרון:
- תקורה: תהליכי הצפנה ופענוח כוללים תקורה חישובית, אשר יכולה להשפיע על ביצועי המערכת.