

Document Design-my_copy.c

- **מטרת התוכנית:**
התוכנית מקבלת קובץ מקור ויעד ומעתקה את התוכן של קובץ המקור לקובץ היעד
- **ביצד עומדת בדרישות:**
התוכנית מקבלת שני ארגומנטים משורט הפוקודה: קובץ מקור וקובץ יעד, ובמצעת בדיקה שמספר הארגומנטים תקין לפני תחילת הביצוע. פתיחת הקבצים מתבצעת באמצעות קריאות מערכת (open) בלבד, בהתאם לדרישת המטלה, תוך טיפול בשגיאות אפשריות. העתקת הנתונים מתבצעת באמצעות לולאה המשמשת בקריאות read ו-write, כך שהתוכן מועתק במקטעים (Buffer) ולא בטיענה מלאה לזכורן. התוכנית כוללת טיפול בשגיאות בכל שלב קריטי: פתיחת קבצים, קריאה וכטיבה, ובמקרה של שגיאה מפסיקה את פעולתה בצורה מבוקרת. בסיום ההעתקה התוכנית סוגרת את כל הקבצים הפתוחים באמצעות close, ובכך מונעת דליפוט משאבים. באופן זה, התוכנית מימושה העתקת קבצים תקינה, יעילה ועומדת בכל דרישות המטלה, תוך שימוש ב-System Calls בלבד.

- **ניתוח Calls System :**

תפקיד בתוכנית	פלט (Return Value)	קלט(Arguments)	שורות בקוד	קריאה מערכת
הdfsota הودעת שגיאה במקורה של מספר ארגומנטים שגוי	מספר בתים שנכתבו / -1	STDERR_FILENO, מחרוזות שגיאה, אורך	14–15	write
בדיקה אם קובץ המקור קיים	0 / -1	נתיב קובץ מקור , F_OK	18–22	access
בדיקה אם קיימת הרשות קריאה לקובץ המקור	0 / -1	נתיב קובץ מקור , R_OK	24–28	access
בדיקה האם קובץ היעד כבר קיים	0 / -1	נתיב קובץ יעד , F_OK	30	access
הציג אזהרה למשתמש במקורה שקובץ היעד קיים	מספר בתים שנכתבו / -1	הודעה למשתמש, fd=1, או רץ	33–35	write
קריאה תשובה המשמש(מ/ע)	מספר בתים שנקרו או (<0, 0 בסוף קובץ, או -1 במקורה של שגיאה	fd=0, כתובות משתנה answer, 1	37–41	read
קריאה חוזרת עד קבלת קלט חוקי	מספר בתים שנקרו או (<0, 0 בסוף קובץ, או -1 במקורה של שגיאה	fd=0, כתובות משתנה answer, 1	45–49	read
פתיחה קובץ המקור לקריאה	fd / -1	נתיב מקור , O_RDONLY	61–66	open
O_TRUNC	O_CREAT	נתיב יעד, O_WRONLY O_CREAT O_TRUNC, 0644	68–75	open
קריאה בלולאה מקובץ המקור	מספר בתים / 10 / -	buffer, fd מ庫ר, BUFFER_SIZE	77	read
כתיבה הנתונים לקובץ היעד	מספר בתים 1 / - /	buffer, יעד fd bytes_read	79–84	write
סגירת קוביizi המקור ויעד	0 / -1	fd	95–96	close

• לוגיקת זרימה (Flow)

פעולות התוכנית מתבצעות בשלבים הבאים:

1. קראית ארגומנטים:

בתחילת הריצה נבדק מספר הארגומנטים שהועברו משורת הפקודה (argc). אם מספר הארגומנטים אינו שווה ל-3 (שם התוכנית, קובץ מקור וקובץ יעד), התוכנית מדפסה הודעה שגיאה ל-stderr ומסיימת את פעולתה.

2. פתיחה קובץ המקור:

התוכנית מנסה לפתוח את קובץ המקור במצב קראיה באמצעות קראית המערכת open(argv[1], O_RDONLY). אם הפתיחה נכשלה (ערך החזרה שלילי), מודפסת הודעה שגיאה מתאימה והתוכנית מסיימת את הריצה.

3. פתיחה / יצירה של קובץ היעד:

לאחר פתיחה מוצלחת של קובץ המקור, התוכנית מנסה לפתוח או ליצור את קובץ היעד באמצעות כתיבה באמצעות O_WRONLY | O_CREAT | O_TRUNC (ערך החזרה של 0644 במקורה של כל, התוכנית סוגרת תחילת את קובץ המקור, מדפסה הודעה שגיאה, ומסיימת את פעולתה בזורה מבוקרת).

4. לולאת העתקה:

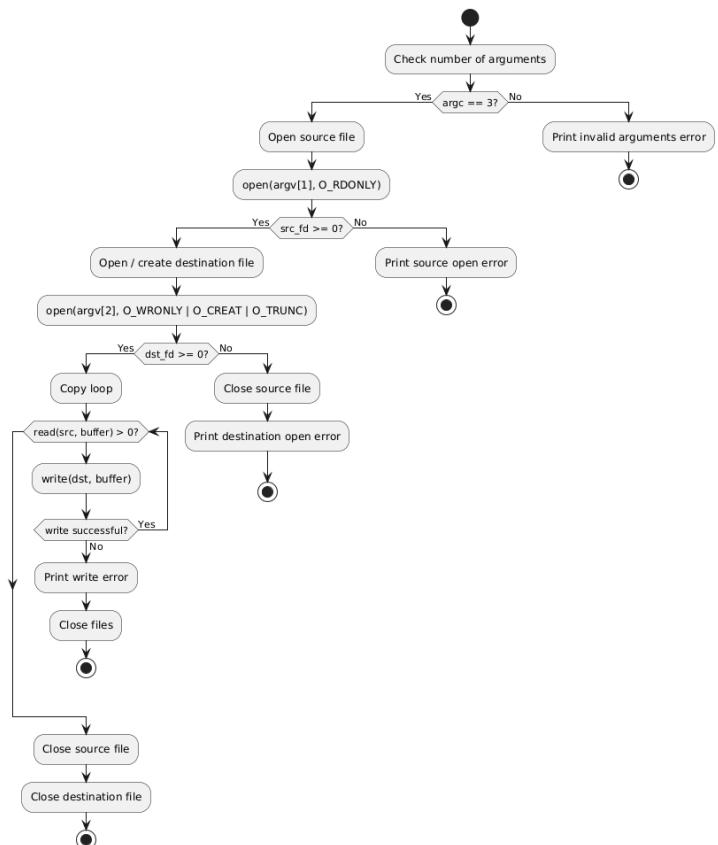
התוכנית נכנסת לולאה שבה ממבצעת קראיה של נתונים מקובץ המקור באמצעות read אל תוך ה-Buffer, ולאחר מכן כתיבה של הנתונים שנקרוואו לקובץ היעד באמצעות write. הלוואה נמשכת כל עוד read מחזירה מספר בתים חיובי. ערך החזרה של אפס מציין הגעה לסוף הקובץ.

5. טיפול בשגיאות במהלך העתקה:

במהלך הלוואה נבדק כי פעולה write הצלילה וכי מספר הבטים שנכתב תואם במספר הבטים שנקרוואו. במקרה של כבכיה, מודפסת הודעה שגיאה, הקבצים הפתוחים נסגורים, והתוכנית מסיימת את פעולתה.

6. סיום וסגירת קבצים:

עם סיום העתקה (הגעה לסוף הקובץ), התוכנית סוגרת את קובץ המקור ואת קובץ היעד באמצעות close, ולאחר מכן מסיימת את פעולה בהצלחה.



- **ניהול Buffer ויעילות:**
בחرتני לעבוד עם Buffer בגודל 4096 בתים, לאחר שזהו גודל בלוק סטנדרטי של מערכת הקבצים ברוב מערכות Unix-Linuks. עבודה עם גודל זה מאפשרת התאמת ישירה לאופן שבו הנדרדים נקראים וככובים בפועל באמצעות מנגנון ה-I/O של מערכת הקבצים.
בחירה ב-Buffer קטן יותר הייתה גורמת לריבוי קריאות למערכת (read), מה שMOVEDIL לעומס מיותר ולפגיעה ביעילות עקב מעבר תכוף בין מרחב המשמש למרחוב הernal. לעומת זאת, שימוש ב-Buffer גדול יותר אינו משפר את הביצועים, משום שמערכת הקבצים ממילא מטפלת ב-I/O בלבד ביחסות של בלוקים בגודל זה, ולכן אין יתרון ממשי בקריאה של כמות גדולה יותר בכלל פעולה.
לכן, גודל Buffer של 4096 בתים מהויה פתרון יעיל ומאוזן, שמקטין את מספר קריאות המערכת, מנצח בצורה מיטבית את מנגנון ה-I/O של Linux, ושומר על קוד פשוט וברור.
בנוסף, מבחינת מספר קריאות המערכת, בכל איטרציה של הלולאה מתבצעת קריאת מערכת אחת מסוג read וקריאה מערכת אחת מסוג write, וכך מספר ה-System Calls תלוי ישירות במספר האיטרציות. עבור קובץ בגודל N בתים ו-Buffer בגודל B , מספר האיטרציות הוא בקירוב $[N/B]$, וכך מספר קריאות המערכת הכלול הוא $O(N/B)$. בחירה ב- $B = 4096$ בתים מקטינה משמעותית את מספר ה-System Calls בהשוואה ל- B קטנים יותר, וכך משפרת את יעילות הפתרון.
- **טיפול בשגיאות:**
התוכנית מבצעת טיפול מפורש בשגיאות של קריאות המערכת (open, read, write, close) על-ידי בדיקת ערך החזרה של כל קריאה. בהתאם למסך של Linux, קריאות מערכת מחזירות ערך שלילי (בדרכם כלל -1) במקרה של שגיאה, וכך לאחר כל קריאה מתבצעת בדיקה האם התוצאה מוצביה על כשל.
במקרה של שגיאה בפתיחת קובץ (open), התוכנית מזזה שהקריאה נכשלה ומדפסה הודעה שגיאה מתאימה ל-stderr, ולא ממשיכה לביצוע פעולות נוספות על קובץ שלא נפתח בהצלחה. באופן דומה, לאחר כל קריאת read נבדק ערך החזרה: ערך שלילי מצביע על שגיאה בקריאה, וערך של אפס מצביע סוף קובץ. גם בקריאה write נבדק כי מספר הבטים שנכתב תואם למספר הבטים שנקבעו, כדי לוודא שלא התרחשה שגיאה בכתיבתם.
בנוסף, התוכנית סוגרת קבצים פתוחים באמצעות close ומטפלת בשגיאות אפשריות גם בשלב זה, על מנת למנוע דיליפות משאבים. גישה זו מבטיחה שהתוכנית תזהה כשלים בזמן אמיתי, תדוח עליהם בצורה ברורה, ותפסיק את פעולתה בצורה מבוקרת במקרה של שגיאת System Call.