

## Document Design-my\_copy.c

- **מטרת התוכנית:**  
התוכנית מקבלת קובץ מקור ויעד ומעתיקה את התוכן של קובץ המקור לקובץ היעד
- **כיצד עומדת בדרישות:**  
התוכנית עומדת בדרישות המטלה באופן הבא:  
התוכנית מקבלת שני ארגומנטים משורת הפקודה: קובץ מקור וקובץ יעד, ומבצעת 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 המשתמשת בקריאות read ו-write, כך שהתוכן מועתק במקטעים (Buffer) ולא בטעינה מלאה לזיכרון.  
התוכנית כוללת טיפול בשגיאות בכל שלב קריטי: פתיחת קבצים, קריאה וכתובה, ובמקרה של שגיאה מפסיקה את פעולתה בצורה מבוקרת.  
בסיום ההעתיקה התוכנית סוגרת את כל הקבצים הפתוחים באמצעות close, ובכך מונעת דליפות משאבים.  
באופן זה, התוכנית מממשת העתקת קבצים תקינה, יעילה ועומדת בכל דרישות המטלה, תוך שימוש ב-System Calls בלבד.

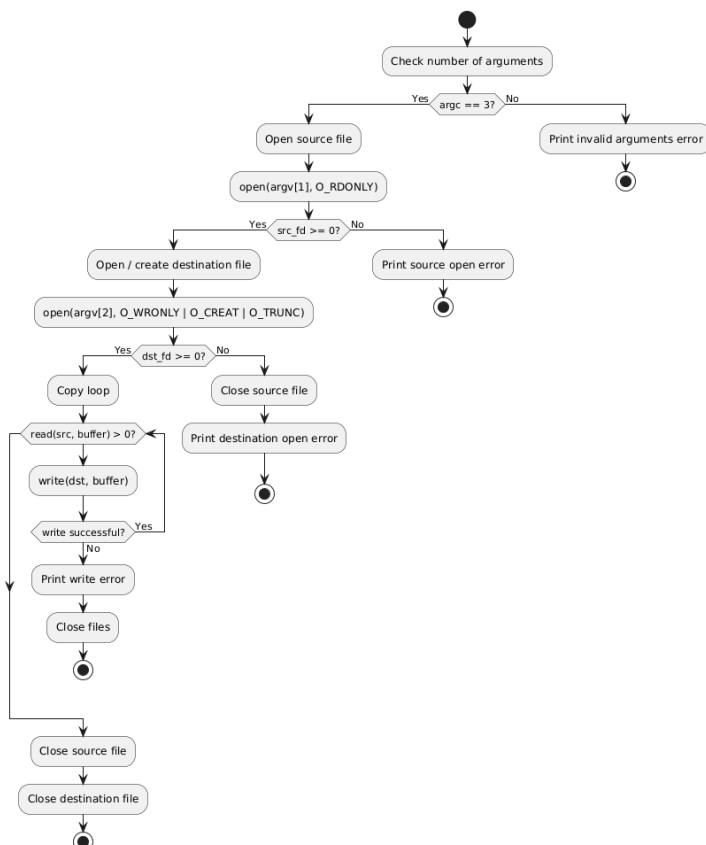
• **ניתוח System Calls:**

קריאת מערכת	שורות בקוד	קלט (Arguments)	פלט (Return Value)	תפקיד בתוכנית
write	14–15	STDERR_FILENO, מחרוזת שגיאה, אורך	מספר בתים שנכתבו / -1	הדפסת הודעת שגיאה במקרה של מספר ארגומנטים שגוי
access	18–22	נתיב קובץ מקור, F_OK	0 / -1	בדיקה אם קובץ המקור קיים
access	24–28	נתיב קובץ מקור, R_OK	0 / -1	בדיקה אם קיימת הרשאת קריאה לקובץ המקור
access	30	נתיב קובץ יעד F_OK, ,	0 / -1	בדיקה האם קובץ היעד כבר קיים
write	33–35	fd=1, הודעה למשתמש, אורך	מספר בתים שנכתבו / -1	הצגת אזהרה למשתמש במקרה שקובץ היעד קיים
read	37–41	fd=0, כתובת משתנה, answer, 1	מספר בתים שנקראו (<0), 0 בסוף קובץ, או -1 במקרה של שגיאה	קריאת תשובת המשתמש (y/n)
read	45–49	fd=0, כתובת משתנה, answer, 1	מספר בתים שנקראו (<0), 0 בסוף קובץ, או -1 במקרה של שגיאה	קריאה חוזרת עד קבלת קלט חוקי
open	61–66	נתיב מקור, O_RDONLY	fd / -1	פתיחת קובץ המקור לקריאה
open	68–75	נתיב יעד, O_WRONLY   O_CREAT   O_TRUNC, 0644	O_CREAT	O_TRUNC`
read	77	fd מקור, buffer, BUFFER_SIZE	מספר בתים / -10 /	קריאה בלולאה מקובץ המקור
write	79–84	fd יעד, buffer, bytes_read	מספר בתים / -1	כתיבת הנתונים לקובץ היעד
close	95–96	fd	0 / -1	סגירת קובצי מקור ויעד

## • לוגיקת זרימה (Flow)

### פעולת התוכנית מתבצעת בשלבים הבאים:

1. קריאת ארגומנטים:  
בתחילת הריצה נבדק מספר הארגומנטים שהועברו משורת הפקודה (argc). אם מספר הארגומנטים אינו שווה ל-3 (שם התוכנית, קובץ מקור וקובץ יעד), התוכנית מדפיסה הודעת שגיאה ל-stderr ומסיימת את פעולתה.
2. פתיחת קובץ המקור:  
התוכנית מנסה לפתוח את קובץ המקור במצב קריאה באמצעות קריאת המערכת `open(argv[1], O_RDONLY)`. אם הפתיחה נכשלת (ערך החזרה שלילי), מודפסת הודעת 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 במצב כתיבה באמצעות `open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0644)` במקרה של כשל, התוכנית סוגרת תחילה את קובץ המקור, מדפיסה הודעת שגיאה, ומסיימת את פעולתה בצורה מבוקרת.
4. לולאת העתקה:  
התוכנית נכנסת ללולאה שבה מתבצעת קריאה של נתונים מקובץ המקור באמצעות `read` אל תוך ה-Buffer, ולאחר מכן כתיבה של הנתונים שנקראו לקובץ היעד באמצעות `write`. הלולאה נמשכת כל עוד `read` מחזירה מספר בתים חיובי. ערך החזרה של אפס מציין הגעה לסוף הקובץ.
5. טיפול בשגיאות במהלך ההעתקה:  
במהלך הלולאה נבדק כי פעולת `write` הצליחה וכי מספר הבתים שנכתב תואם למספר הבתים שנקראו. במקרה של כשל בכתיבה, מודפסת הודעת שגיאה, הקבצים הפתוחים נסגרים, והתוכנית מסיימת את פעולתה.
6. סיום וסגירת קבצים:  
עם סיום ההעתקה (הגעה לסוף הקובץ), התוכנית סוגרת את קובץ המקור ואת קובץ היעד באמצעות `close`, ולאחר מכן מסיימת את פעולתה בהצלחה.



- **ניהול Buffer ויעילות:**

בחרתי לעבוד עם Buffer בגודל 4096 בתים, מאחר שזהו גודל בלוק סטנדרטי של מערכת הקבצים ברוב מערכות Linux. עבודה עם גודל זה מאפשרת התאמה ישירה לאופן שבו הנתונים נקראים ונכתבים בפועל באמצעות מנגנון ה-I/O של מערכת הקבצים. בחירה ב-Buffer קטן יותר הייתה גורמת לריבוי קריאות מערכת (read), מה שמוביל לעומס מיותר ולפגיעה ביעילות עקב מעבר תכוף בין מרחב המשתמש למרחב הקרנל. לעומת זאת, שימוש ב-Buffer גדול יותר אינו משפר את הביצועים, משום שמערכת הקבצים ממילא מטפלת ב-I/O ביחידות של בלוקים בגודל זה, ולכן אין יתרון ממשי בקריאה של כמות גדולה יותר בכל פעולה.

לכן, גודל Buffer של 4096 בתים מהווה פתרון יעיל ומאוזן, שמקטין את מספר קריאות המערכת, מנצל 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 וקריאת מערכת אחת מסוג write, ולכן מספר ה-System Calls תלוי ישירות במספר האיטרציות. עבור קובץ בגודל  $N$  בתים ו-Buffer בגודל  $B$ , מספר האיטרציות הוא בקירוב  $\lceil N/B \rceil$ , ולכן מספר קריאות המערכת הכולל הוא  $O(N/B)$ . בחירה ב-Buffer בגודל 4096 בתים מקטינה משמעותית את מספר ה-System Calls בהשוואה ל-Buffers 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 ההחזרה של כל קריאה. בהתאם לממשק של Linux, קריאות מערכת מחזירות ערך שלילי (בדרך כלל -1) במקרה של שגיאה, ולכן לאחר כל קריאה מתבצעת בדיקה האם התוצאה מצביעה על כשל.

במקרה של שגיאה בפתיחת קובץ (open), התוכנית מזהה שהקריאה נכשלה ומדפיסה הודעת שגיאה מתאימה ל-stderr, ולא ממשיכה לביצוע פעולות נוספות על קובץ שלא נפתח בהצלחה. באופן דומה, לאחר כל קריאת read נבדק ערך ההחזרה: ערך שלילי מצביע על שגיאה בקריאה, וערך של אפס מציין סוף קובץ. גם בקריאת write נבדק כי מספר הבתים שנכתב תואם למספר הבתים שנקראו, כדי לוודא שלא התרחשה שגיאה בכתיבה.

בנוסף, התוכנית סוגרת קבצים פתוחים באמצעות close ומטפלת בשגיאות אפשריות גם בשלב זה, על מנת למנוע דליפות משאבים. גישה זו מבטיחה שהתוכנית תזהה כשלים בזמן אמת, תדווח עליהם בצורה ברורה, ותפסיק את פעולתה בצורה מבוקרת במקרה של שגיאת System Call.