הרצוג

GIOS

גיא קורן

216870972

יוסף חי זהבי

הגנת סייבר ומערכות הפעלה

מבוא

**תקציר ורציונל לפרויקט GIOS**

GIOS היא מערכת הפעלה בסיסית שנבנית מאפס במטרה לספק הבנה מעמיקה במבני ליבה של מערכות הפעלה. המערכת כוללת תמיכה בקלט מהמקלדת, הצגת מידע למסך, ניהול זיכרון דינמי, מעטפת פקודות (Shell) בסיסית, מערכת סידור קבצים פשוטה, וריבוי תהליכים. בנוסף, המערכת תוכל לפעול לצד שרת, שיאפשר למשתמשים להתחבר אליה מרחוק ולהתקשר עם המערכת דרך לקוח.

**מטרת המוצר המוגמר**

המערכת תספק למשתמש אפשרות להרים לקוח שיתחבר למערכת הפעלה הרצה על שרת, ותאפשר אינטראקציה בסיסית עם המערכת. מטרת המוצר המוגמר היא לאפשר למשתמש לעבוד עם מערכת הפעלה מבוססת CLI (ממשק שורת פקודה) ולבצע פעולות שונות כמו ניהול קבצים, הרצת תהליכים ותחזוקת המערכת, תוך כדי שימוש בקלט/פלט ממקלדת ומסך.

**רציונל לבחירת הפרויקט**

הבחירה בפרויקט נועדה לאפשר העמקה בנושאים כגון ניהול משאבים, תפעול חומרה, תכנות ברמה נמוכה ותכנון מבנה מערכת הפעלה. בניית מערכת הפעלה מאפס תספק לי הבנה יסודית של מבנה המעבד וניהול הזיכרון. פרויקט זה גם יאפשר לי לפתח את המיומנויות שלי בתכנות ב-C ו-Assembly, ולהתנסות באתגרים טכניים הנוגעים לפיתוח מערכות הפעלה עצמאיות.

**אתגרים צפויים בפרויקט**

בפרויקט צפויים מספר אתגרים מרכזיים:

1. **ניהול זיכרון דינמי בצורה יעילה** – פיתוח מערכת ניהול זיכרון שתתמוך בתהליכים מרובים בצורה אופטימלית.
2. **עבודה ישירה עם חומרה** – הבנת קריאות מערכת ברמה נמוכה והבקרה על חומרה כמו המקלדת והמוניטור.
3. **הפעלת המערכת מרחוק** – יציבות וביטחון בקישוריות מרחוק וביצוע פעולות במערכת הפעלה דרך חיבור רשת.
4. **ריבוי תהליכים** – ניהול תהליכים במקביל והקצאת משאבים בין תהליכים בצורה מאורגנת ויעילה.

האתגרים הללו מציבים דרישות גבוהות עבור יציבות וביצועים, ומספקים הזדמנות לפיתוח יכולות טכניות בתחום מערכות ההפעלה.

המערכת מיועדת לאנשים שרוצים להתנסות עם kernel פשוט ואו רוצים להריץ את המערכת מרחוק בשביל לאפשר גישה למחשב שלך שכולל קבצים וכו מבלי להיות פיזית ליד המחשב.  
  
  
  
המערכת שואפת להציג את היכולת לפתח מערכת הפעלה עם כלים בסיסיים ומתקדמים, והיא כוללת את המטרות המרכזיות הבאות:

1. **ניהול זיכרון תהליכים מרובים (multi-process memory management)** – ניהול זיכרון עבור מספר תהליכים במקביל.
2. **יכולת כתיבה וקריאה מקבצים** – התמחות בכתיבה, קריאה, ארגון קבצים ויצירת ספריות.
3. **מעטפת פקודות בסיסית (shell)** – יצירת מעטפת פקודות המאפשרת למשתמש לתקשר עם המערכת.
4. **גישה מרחוק (remote access)** – מתן אפשרות להתחבר למערכת מרחוק ולבצע פעולות שונות

**בעיות, תועלות וחסכונות**

הבעיה העיקרית היא פיתוח מערכת הפעלה ברמה נמוכה שמתמודדת עם אתגרים של ניהול זיכרון, עבודה עם חומרה, ואספקת יכולת גישה מרחוק. המטרה היא ליצור מערכת הפעלה פשוטה ויעילה שתאפשר ניהול של תהליכים, קבצים ונתונים בצורה ממוחשבת ונגישה. התועלות הצפויות כוללות:

* יכולת ניהול תהליכים בצורה אפקטיבית ומודולרית.
* ניהול קבצים וספריות באופן פשוט ונגיש.
* גישה מרחוק שתספק למשתמשים גמישות בביצוע פקודות והפעלת המערכת. השירותים המערכת תספק כוללים:
* ניהול זיכרון ודיסק.
* אפשרות לעבוד עם קבצים ודירוגם.
* ביצוע פעולות באמצעות מעטפת פקודות.
* חיבור מרחוק למערכת.

**סקירת פתרונות קיימים**

המערכת GIOS מציעה פתרון ייחודי בכך שהיא משלבת יכולות של ניהול זיכרון, קבצים וכניסות מרחוק בגרסה פשוטה אך עוצמתית של מערכת הפעלה. מערכות הפעלה כמו Linux ו-Windows מציעות פתרונות רחבים יותר, אך המערכת GIOS מיועדת במיוחד ללמוד ולפתח עקרונות בסיסיים של ניהול תהליכים, קבצים וזיכרון, ובכך מציעה גישה אחרת לעבודה עם חומרה ברמה נמוכה.  
  
**סקירת טכנולוגית הפרויקט**

הטכנולוגיות בהן נעשה שימוש בפרויקט הן מוכרות ומיועדות למערכות הפעלה ברמה נמוכה – שפות C ו-Assembly, כולל תכנון עצמאי של ניהול זיכרון וקבצים. הטכנולוגיה אינה חדשה, אך המערכת מתמודדת עם אתגרים כמו ניהול תהליכים מרובים וכתיבה לקריאה של קבצים במערכת הפעלה עצמאית. ישנן הגבלות טכנולוגיות כמו תמיכה מינימלית בחומרה ובספריות חיצוניות.  
  
**תיחום הפרויקט**

המערכת עוסקת בניהול זיכרון, ניהול קבצים, מעטפת פקודות ויכולת גישה מרחוק. התחומים בהם המערכת אינה עוסקת כוללים אבטחת מידע, תמיכה במערכות רשת מורכבות, ומערכות הפעלה עם ניהול זיכרון משוכלל יותר כמו virtual memory.   
  
  
**תיאור מפורט של המערכת:**

מערכת ההפעלה GIOS תספק ממשק למשתמש ברמת ליבה, ותכלול תמיכה ביכולות בסיסיות כמו קלט ממקלדת, פלט למסך, ניהול זיכרון דינמי, מערכת סידור קבצים, ושימוש בתהליכים מרובים. המערכת תהיה מבוססת CLI (ממשק שורת פקודה), ותאפשר למשתמש לבצע פעולות בסיסיות על המערכת דרך שורת פקודה, בנוסף ללקוח יהיה ממשק GUI בפייטון.

המערכת תאפשר את הפונקציות הבאות:

* **ניהול זיכרון דינמי** – ניהול יעיל של הזיכרון עבור מספר תהליכים במקביל.
* **הצגת פלט למסך** – הצגת מידע על המסך בממשק פשוט אך פעיל.
* **קלט מהמקלדת** – קבלת קלט מהמשתמש באמצעות המקלדת.
* **מערכת סידור קבצים** – יצירה, קריאה וכתיבה של קבצים, ארגון קבצים בספריות.
* **תמיכה בתהליכים מרובים** – הרצת מספר תהליכים בו זמנית תוך שמירה על סדר ויעילות.
* **גישה מרחוק** – חיבור מרחוק למערכת הפעלה דרך לקוח המתקשר עם השרת.

**פירוט היכולות לכל סוג משתמש במערכת:**

1. **משתמש רגיל (End User):**
   * הרצת פקודות Shell בסיסיות.
   * יצירה, קריאה, עדכון ומחיקת קבצים.
   * ניווט בין ספריות.
   * גישה למידע תהליכים רצים.
   * חיבור למערכת דרך לקוח מרוחק לצורך אינטראקציה עם המערכת.
2. **מנהל מערכת (Admin):**
   * ניהול משאבים (זיכרון, תהליכים).
   * ניהול קבצים, תיקיות וזכויות גישה.
   * היכולת לעדכן, להפסיק ולהתחיל תהליכים.
   * ניהול חיבורים מרחוק למערכת.
   * ניטור המערכת וביצועים בזמן אמת.

**פירוט הבדיקות (black box):**

בדיקות קופסה שחורה מתמקדות בהערכת ההתנהגות של המערכת לפי דרישות המשתמש, ללא בחינה של הקוד הפנימי:

1. **בדיקת קלט/פלט ממקלדת למסך:**
   * **מטרה**: לבדוק אם המערכת מקבלת קלט נכון מהמקלדת ומציגה את הפלט כראוי למסך.
   * **ביצוע**: הרצת פקודות קלט שונות ועריכת יציאות שונות למסך. צפייה בהדפסת התוצאה למסך והערכה אם הפלט תואם לקלט.
2. **בדיקת ניהול קבצים:**
   * **מטרה**: לבדוק אם המערכת תומכת ביצירה, קריאה, עדכון ומחיקת קבצים.
   * **ביצוע**: יצירת קבצים, כתיבה וקריאה מהם, מחיקת קבצים ובדיקת קיום הקבצים לאחר הפעולות.
3. **בדיקת ניהול תהליכים:**
   * **מטרה**: לבדוק אם המערכת מסוגלת להריץ תהליכים מרובים ולנהל אותם בצורה נכונה.
   * **ביצוע**: הרצת מספר תהליכים בו זמנית, תיעוד ההתנהגות והבטחת ביצועים תקינים של כל תהליך.
4. **בדיקת גישה מרחוק:**
   * **מטרה**: לבדוק אם ניתן להתחבר למערכת דרך לקוח מרוחק.
   * **ביצוע**: חיבור לשרת מרחוק, הרצת פקודות ותצפית על ביצועים ועל יציבות החיבור.
5. **בדיקת יציבות המערכת:**
   * **מטרה**: לבדוק יציבות המערכת תחת עומס תהליכים וביצועים.
   * **ביצוע**: הרצת תהליכים בו זמנית, עם צריכת זיכרון גבוהה, והערכת יציבות המערכת.

**תכנון וניהול לו"ז זמנים לפיתוח המערכת:**

השלבים העיקריים בפרויקט כוללים את הפיתוח הבסיסי של המערכת, פיתוח תהליכים, פיתוח מערכת סידור קבצים, והוספת היכולת להתחבר מרחוק. הלו"ז יכול להתחלק לשלבים הבאים:

1. **שלב ראשון: פיתוח ליבת המערכת (חודש 1-2):**
   * פיתוח ניהול זיכרון דינמי.
   * פיתוח קלט/פלט ממקלדת ומסך.
   * תכנון בסיסי של מבנה המערכת.
2. **שלב שני: פיתוח מערכת סידור קבצים (חודש 2-3):**
   * יצירה וכתיבה לקריאה מקבצים.
   * ארגון הקבצים בספריות.
3. **שלב שלישי: פיתוח ריבוי תהליכים (חודש 3-4):**
   * פיתוח המודול של ניהול תהליכים.
   * בדיקות ייצוב וביצועים תחת עומס.
4. **שלב רביעי: חיבור מרחוק (חודש 4-5):**
   * פיתוח תמיכה בחיבור מרחוק.
   * בדיקות חיבור וביצועים.
5. **שלב חמישי: בדיקות ותחזוקה (חודש 5-6):**
   * הרצת בדיקות קופסה שחורה (black box) ותחזוקה של המערכת.

**ניהול הסיכונים בפרויקט:**

1. **סיכון 1: בעיות ביציבות המערכת תחת עומס (ניהול תהליכים/זיכרון):**
   * **דרכים להתמודדות** : ייבוצע אופטימיזציה של ניהול הזיכרון, תהליכים יבוקרו ויוסרו אם לא נדרשים.
   * **ביצוע בפועל**: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. **סיכון 2: קשיים בתמיכה בחיבור מרחוק:**
   * **דרכים להתמודדות**: תכנון מוקדם של פרוטוקול תקשורת יציב ושימוש ביכולות אבטחה בסיסיות.
   * **ביצוע בפועל**: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
3. **סיכון 3: בעיות בתמיכה במערכת סידור הקבצים:**
   * **דרכים להתמודדות**: תכנון יעיל של מנגנוני קובץ עם אפשרויות לקריאה, כתיבה ומחיקה בצורה אופטימלית.
   * **ביצוע בפועל:** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*