

נושא הפרויקט: **Microkernel**

שם התלמיד: **עדן פרנקל**

ת.ז. של התלמיד: **206964124**

שם בית הספר והעיר: **אורט גוטמן, נתניה**

שמות המנחים והמורים המלווים: **מיכאל צ'רנובילסקי וגד רוזנטל**

מועד הגשת המסמך: **10.02.2016**

**תיאור המוצר המוגמר**

שם המוצר המוגמר הוא EdenOS, והוא מערכת הפעלה מצומצמת.

המערכת ממששת:

* Boot and Kernel Loading
* Memory Management
* Interrupt Handling
* Multi-Tasking & Task Management
* User Interface
* File System
* Device Management (בסיסי)

המטרה שלי בעבודה זו היא ליצור מערכת עובדת שתוכל לנהל ברמה בסיסית את משאבי החומרה והתוכנה הבסיסיים של המחשב, ותוכל לספק למשתמש ממשק טקסטואלי שבו הוא יוכל להזין פקודות ולעשות שימוש בפעולות המערכת.

מערכת ההפעלה שלי תסייע למשתמש להשתמש במשאבי המחשב שלו, לעבוד בסביבה וירטואלית שתספק לו שירותים בסיסיים באמצעות פקודות פשוטות.

**תיאור הפיתוח**

במהלך הכנת הפרויקט ביצעתי תהליך מעמיק של מחקר בתחומים שונים, החל מתחום החומרה ועד לתחום התוכנה. תחילה, הייתי צריך לחקור כיצד עובד תהליך ה-Boot במחשב על מנת שאוכל לכתוב קוד שיממש את התהליך הזה ויטען את מערכת ההפעלה שלי לזיכרון המחשב. בתהליך מחקר זה נעזרתי בעיקר באתר האינטרנט wiki.osdev.org המציע ערכים שונים לגבי תחומים מגוונים בפיתוח מערכת הפעלה, כולל ה-boot sequence. בנוסף, נעזרתי בפורומים שונים בהם דיווחו אנשים שונים על בעיות דומות שבהן אני נתקלתי בתהליך, ונעזרתי בתשובות שהם קיבלו מאנשים מנוסים יותר. את חלק זה של המערכת כתבתי ב-Assembly, שפה אשר למדתי בכיתה י', ולכן לא היה לי צורך לבצע מחקר מעמיק לגבי השימוש בשפה, אלא רק לגבי דברים ספציפיים שלא הכרתי.

לאחר מכן, הייתי צריך לפתח את מערכת ההפעלה עצמה בשפת C, שפה שמעולם לא התנסיתי. תחילה, ניגשתי לאתרים שונים המציעים ללמד את השפה. לאחר מכן, נעזרתי בספרים שונים על שפת C שנתקלתי בהם באינטרנט, כמו " The C Programming Language".

לפני שהתחלתי ממש לכתוב קוד, החלטתי לקרוא קצת לגבי המבנה של מערכות הפעלה, כדי שיהיה לי קצת רקע. מלבד אתרים שונים למתכנתי מערכות הפעלה שמציעים אלפי מאמרים לגבי תחומים שונים, נתקלתי גם בספרים שונים, לדוגמה, "Operating Systems Design and Implementation" שנכתב ע"י יוצר מערכת ההפעלה MINIX שעליה התבססה מערכת ההפעלה Linux. ספר זה נתן לי ידע ראשוני וידע בסיסי לגבי אין בנויה מערכת ההפעלה, וכיצד היא עובדת מאחורי הקלעים.

לאחר מכן, היה לי צורך ב-compiler שימיר את הקוד שלי לקוד בינארי שיוכל לרוץ על המאבד. חקרתי וקראתי כיצד "לבנות" compiler מותאם לצרכים שלי ולמעבד שעליו אני עובד.

מרבית המחקר שביצעתי לאחר מכן היה לגבי תחומים ספציפיים יותר במערכת ההפעלה, כמו ניהול זיכרון, טיפול בפסיקות ועוד. בשביל כל התחומים האלו נעזרתי בשני אתרים עיקריים המיועדים למתכנתי מערכות הפעלה (wiki.osdev.org ו-www.osdever.net) המכילים מאמרים רבים ואף מדריכים כיצד לממש תכונות שונות של מערכת ההפעלה, ופירוט על דרך הפעולה של מערכת ההפעלה וגישות שונות לגבי תכונות מסוימות שלה.

ראוי לציין שבמקרים מסוים פניתי (או הופניתי) ע"י מאמרים שונים ל-Manuals רשמיים של המעבד, שם קראתי בפירוט כיצד יש לנצל תכונות מסוימות שלו ולממש באמצעותו דברים מסוימים.

בהמשך אני מתכנן להמשיך את המחקר לגבי תחומים שעוד לא מימשתי, ולהעמיק את המחקר בתחומים מסוימים במידת הצורך. לדוגמה, כעת עלי ללמוד לעומק כיצד פועלת מערכת הקבצים שאותה אני מתכוון לממש (FAT), על מנת שאוכל לממש אותה בצורה טובה.

**תיאור המודולים**

ראשית, ברצוני להציג מבט-על על המודולים השונים ויחסי הגומלין ביניהם.

תיאור פשוט של תהליך ה-Boot ושל טעינת ה-Kernel:

תיאור כללי של התקשורת בין חלקי המערכת העיקריים (שיתוף הפעולה ביניהם):

User Interface

Interrupts Handling

File System

Task Management

Device Management

Memory  
Management

* **Memory Management –** רכיב זה של המערכת מנהל את הזיכרון ומקצה לתהליכים שונים שטח בזיכרון. רכיב זה אינו נעזר בשום רכיב אחר של המערכת, אך רכיבים רבים נעזרים בו על מנת לתפקד כראוי.
* **Task Management –** רכיב זה של המערכת ינהל את התהליכים השונים ויממש Scheduler. רכיב זה נעזר ברכיבי המערכת הבאים:
  + Memory Management – על מנת להקצות ולשחרר שטחים בזיכרון לתהליכים השונים.
  + Interrupts Handling – על מנת לקבל התראות על הזמן שעובר (ממומש באמצעות ה-RTC) למימוש ה-Scheduler.
* **Interrupts Handling –** רכיב זה של המערכת אחראי לקבל ולטפל בפסיקות חומרה ותוכנה של המערכת. רכיב זה אינו נעזר בשום רכיב אחר של המערכת, אך רכיבים רבים נעזרים בו על מנת לתפקד כראוי.
* **Device Management –** רכיב זה של המערכת ינהל את ההתקנים השונים (החיצוניים) שיחוברו למערכת. רכיב זה נעזר ברכיבי המערכת הבאים:
  + Interrupts Handling – על מנת לקבל התראות על חיבור/ניתוק של התקנים חיצוניים.
  + File System – על מנת לנתח מערכות קבצים הקיימות על התקנים ניידים ולחברן למערכת הקבצים הראשית של המערכת (Mounting).
* **File System –** רכיב זה של המערכת ינהל את מערכת הקבצים של המערכת, ויאפשר ביצוע פעולות עליה. רכיב זה נעזר ברכיבי המערכת הבאים:
  + Memory Management – על מנת להקצות מקום בזיכרון ולטעון לשם את התוכן הדרוש.
  + Task Management – על מנת להריץ תוכנות השמורות בזיכרון, ועל מנת להפעיל תהליכים ברקע שינהלו את מערכת הקבצים באמצעות ה-Scheduler.
  + Device Management – על מנת לתקשר עם התקנים ניידים באמצעות אבסטרקציה ולקבל את תוכנם (אם הם מכילים מערכת קבצים כלשהי).
* **User Interface –** רכיב זה של המערכת ינהל את כל התקשורת עם המשתמש באמצעות CLI. רכיב זה נעזר ברכיבי המערכת הבאים:
  + Task Management – על מנת לאפשר למשתמש לנהל את התהליכים הרצים במערכת (לדוגמה, הפסקת ריצה של תהליך).
  + Interrupts Handling – על מנת לאפשר את התקשורת עם המשתמש יש לקבל קלט מהמקלדת שמתקבל באמצעות פסיקות חומרה. בנוסף באמצעות רכיב זה ניתן להציג למשתמש נתונים נוספים, כמו שעה או הזמן שחולף.
  + Device Management – על מנת לאפשר למשתמש לנהל את ההתקנים המחוברים למערכת.
  + File System – על מנת לאפשר למשתמש לנהל את מערכת הקבצים ולבצע פעולות עליה (יצירת קובץ, מחיקת קובץ ועוד).

**קשיים**

במהלך הפרויקט נתקלתי בקשיים שונים. חלק מהקשיים נבעו מעובדה שהתחום שבו עוסק הפרויקט שלי הוא מורכב, ולא תמיד קל למצוא את המענה הדרוש לבעיה שצצה.

קשיים אחרים נבעו מהעובדה שאני מתעסק באופן ישיר עם החומרה. עד כה התרגלתי לעבוד עם אבסטרקציה שחסכה ממני את העבודה המפרכת שמקיימת מערכת ההפעלה בתקשורת עם החומרה. כעת, הייתי צריך לקיים תקשורת זו בעצמי, והחומרה לא תמיד הגיבה בדרך שבה ציפיתי או בדרך שבה היא הייתה אמורה לפעול.

קושי נוסף הוא המחקר הרב שנדרשתי לעשות. אומנם למדתי בבית הספר על מערכות הפעלה, אך כמעט ולא ידעתי כלום לגבי מימושן. הייתי צריך לבצע מחקר מעמיק ביותר ולהכניס לנושאים מורכבים שדרשו ממני חשיבה מורכבת ועמוקה, ולאחר מכן לממש את אשר למדתי.

קושי נוסף היה איתור תקלות. בניגוד לתכנות על גבי מערכות הפעלה קיימות, כאן לא היה לי Debugger, והייתי צריך להבין לבד (ובאמצעות פלט בלבד וללא שום כלי ניתוח) היכן נמצאות התקלות שלי וכיצד לסדר אותן. דבר זה גרם לעיכובים ותסכול במקרים מסוימים, אך גם לסיפוק אדיר כשהצלחתי לפתור את הבעיה ולממש את מה שניסיתי לממש.

**ניהול פרויקט עתידי**

|  |  |
| --- | --- |
| **יעד (מה יש לבצע)** | **זמן ביצוע (מתאריך עד תאריך)** |
| Boot, פלט למסך, קלט מהמקלדת, טיפול ב-Interrupts, הרצת קוד בסיסי של Command-Line עם מספר פקודות בסיסיות. | *בוצע* |
| ניהול הזיכרון, Multitasking | *בוצע* |
| מימוש מערכת קבצים | *עד 09.03.2016* |
| תוספות (אם יהיו), תיקוני באגים ויצירת התוצר הסופי | *09.03.2016-06.04.2016* |