# MicroKernel

## מסמך עיצוב

## עדן פרנקל

גרסה 1

06.12.2015

|  |  |  |
| --- | --- | --- |
| תאריך | גרסה | תקציר השינויים |
| 14.12.2015 | 1 | הגשת המסמך הראשוני. |
|  |  |  |
|  |  |  |
|  |  |  |



**הקדמה**

**מטרה**

מטרת מסמך העיצוב היא לתאר את המבנה הפנימי של מערכת הפרויקט, כדי שיהיה למורים, לבוחנים ולי קל יותר להבין כיצד היא בנויה וכיצד היא עובדת מבפנים.

**המוצר**

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, ולעבוד בסביבה וירטואלית שתספק לו שירותים בסיסיים באמצעות פקודות פשוטות.

**קישור למסמכים קודמים**

[מסמך הצעת ותיאור הפרויקט](http://1drv.ms/1QnV6Y9)

**הגדרות**

Microkernel (מיקרו-ליבה) – מערכת הפעלה מצומצמת בעלת ממשק פשוט המנהלת את משאבי המערכת.

RTC – רכיב ה-Real-Time Clock של המחשב, רכיב המשמש למעקב אחר הזמן שחולף.

CLI – Command Line Interface.

**ארכיטקטורת המערכת**

**מבט על**

תיאור פשוט של תהליך ה-Boot וטעינת המערכת:

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

User Interface

Interrupts Handling

File System

Task Management

Device Management

Memory  
Management

**פירוט רכיבי המערכת**

המערכת מורכבת ממספר רכיבים, והעיקריים שבהם הם:

* **Memory Management –** רכיב זה של המערכת ינהל את הזיכרון ויקצה לתהליכים שונים שטח בזיכרון. רכיב זה נעזר ברכיבי המערכת הבאים:
  + Task Management – על מנת לאפשר תהליך שינהל ברקע את הזיכרון באמצעות Scheduling.
* **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.
  + Task Management – על מנת לאפשר תהליך שינהל ברקע את ההתקנים באמצעות ה-Scheduler.
  + 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 (לדוגמא, ניתוק התקן מהמערכת).
  + File System – על מנת לאפשר למשתמש לנהל את מערכת הקבצים ולבצע פעולות עליה (יצירת קובץ, מחיקת קובץ ועוד).

**דיון בנושא העיצוב הנבחר**

בחרתי דווקא בחלוקה זו של רכיבי המערכת משום שהיא מודולרית, מאפשרת אבסטרקציה של רכיבי המערכת השונים ומאפשרת תקשורת נוחה ביניהם. בנוסף, שאבתי השראה ממערכות הפעלה קיימות וצורת העיצוב שלהן.

את תהליך ה-Boot עצמו החלטתי לכתוב ב-Assembly, משום ששם יש צורך בפעולות חומרה בסיסיות שלא מתאפשרות בשפת תכנות עילית כמו C, ויש צורך בשפת סף. את ה-Bootstrap (הקריאה ל-Main Kernel) החלטתי לכתוב גם כן ב-Assembly, על מנת לגשר בין ה-Assembly לקוד ב-C בצורה חלקה.

את ה-Kernel עצמה החלטתי לכתוב ב-C, משום שזו שפה עילית הקרובה יחסית לרמת המכונה, מאפשרת גישה ישרה לזיכרון, ועם זאת מאפשרת יצירה של מבני נתונים ואבסטרקציה מסוימת.

**עיצוב נתונים ופרוטוקולים**

**מערכת הקבצים**

מערכת הקבצים תהיה מסוג FAT (File Allocation Table), ובפרט גרסת FAT32 של מערכת קבצים זו. מערכת הקבצים FAT פותחה ב-1977, והייתה מערכת הקבצים הראשית של מערכות ההפעלה MS-DOS ו-Windows 9x עד החלפתה ב-NTFS במערכת ההפעלה Windows NT. מערכת קבצים זו היא מערכת קבצים פשוטה וחזקה. היא מציעה ביצועים טובים אפילו במימושים פשוטים יחסית.

חלקים עיקריים במערכת הקבצים:

|  |  |
| --- | --- |
| אזור המכיל קוד מכונה האמור להטען לזיכרון על ידי המערכת המובנית של המחשב. | **Boot Sector** |
| אזור המכיל מידע על מערכת הקבצים, נועד על מנת להאיץ פעולות מסוימות על מערכת הקבצים. הוצג לראשונה ב-FAT32. | **FS Information Sector** |
| מערך שבו כל אינדקס מייצג מקטע (Cluster), כל תא מצביע על התא המכיל את החלק הבא של הקובץ, וסוף הקובץ מצוין באמצעות ערך מספרי מיוחד הנמצא מחוץ לטווח. ב-FAT32 גודל כל תא הוא 28 ביטים. | **File Allocation Table(s)** |
| מקטע הנתונים הוא החלק בדיסק שבו נמצאים כל הקבצים והתיקיות. קובץ עשוי להיות מורכב ממספר חלקים שאינם רציפים, אשר ניתן לגשת אליהם באמצעות מעקב סדרתי אחר טבלת הקבצים. | **Data Region** |

**קבצי הרצה**

המערכת תתמוך בשני סוגים של קבצי הרצה – Binary (קוד מכונה פשוט) ו-ELF  
(Executable and Linkable Format). הקבצים הבינאריים הם פשוט קוד מכונה שיועתק באופן לינארי לזיכרון ויורץ. קבצי ה-ELF יורצו באופן דינאמי יותר בהתאם לפורמט.

**ממשק משתמש**

ממשק המערכת יהיה CLI (Command Line Interface), שבו יוכל המשתמש להזין פקודות על מנת לבצע פעולות שונות במערכת, בדומה ל-MS-DOS. במרכז המסך יהיה האזור המרכזי בו יוצגו הקלט והפלט של הפקודות, בחלק העליון של המסך יוצג מידע על התוכנה שרצה כעת (לדוגמה: שם) ובתחתית המסך יוצג מידע על המערכת (לדוגמה: זמן, כמות תהליכים רצים ועוד).