באופן כללי, כנראה נרצה להחזיק כמה משתנים בכל אחד מ-N אובייקטי הקובץ (file object), בשדה ה-private\_data, כאשר N הוא מספר המשחקים המקסימלי שמקבלים כקלט ל-init\_module. כנראה נרצה:

* semaphore לשחקן(ים) הראשון(ים) – מטרתם להבין האם השחקן הראשון הצטרף למשחק
* semaphore לשחקן(ים) השני(ים) – מטרתם להבין האם השחקן השני הצטרף למשחק
* semaphore שננעל ע"י השחקן הלבן (בתורו) ומקבל סיגנלים רק מהשחקן השחור, זאת כדי שאם השחקן הלבן ינסה לבצע 2 מהלכים ברצף, בניסיון השני הוא יאלץ לחכות לסיגנל מהשחור, שיגיע רק בסיום תורו.
* semaphore שננעל ע"י השחור ונפתח ע"י הלבן
* לוח משחק
* מצב המשחק
* מספר MINOR

**הערה:** יהיו הרבה מקומות שנצטרך לשלוח סיגנלים לכל מיני סמפורים כדי למנוע דדלוק. למשל, אם השחקן הלבן ממתין לתורו, ואז השחקן השחור מפסיד/יוצא...

אסור שזה יהיה "גורף"... אם שחקן לבן ממתין לבצע מהלך ואז שחור יוצא, אסור לתת ללבן להמשיך כרגיל...

**שאלות:**

* מה עושים אם מישהו עושה release לפני open?
* נניח המשחק נגמר בניצחון של הלבן, ואז מישהו מהשחקנים יוצא (release). האם לשחקן השני מותר לקרוא את הלוח? מה צריך להחזיר בבדיקת מצב המשחק?
* איך בכלל מזהים מי השחקן? אי אפשר פשוט לשמור pid ב-file\_object כי אם תהליך שעשה open עושה אח"כ fork אז גם הוא וגם הבן שלו הם אותו שחקן

**מקרים לבדיקה:**

קריאה:

* הרבה מנסים לקרוא את מצב הלוח בו זמנית
* הרבה מנסים לקרוא בזמן שגם הרבה כותבים
* הרבה קוראים כשגם הרבה עושים ioctl (זה קורה בכל מקרה בטסטים קודמים, אני יורד מזה)
* בקריאה של 0 תווים מחזירים 0 בלי לעשות כלום
* בקריאה של n תווים (פחות מגודל המטריצה), מחזירים בדיוק n תווים (בלי NULL בסוף)
* בקריאה של גודל המטריצה (96) מחזירים את כל המטריצה
* בקריאה של יותר מ-96 תווים מחזירים בדיוק את המטריצה
* הרבה קוראים בזמן שכמה משחררים את המשחק. צריך לוודא שמי שכן מצליח לקרוא קורא את הכל, ולא חלק / ערכי זבל
* לוודא שקריאה אחרי סגירת המשחק נכשלת
* לוודא שקריאה עם NULL בבאפר מחזיר EFAULT (פיאצה 429)

כתיבה:

* לוודא שרואים תוצאת מהלך בקריאת הלוח
* לוודא שאפשר לקרוא גם אחרי הפסד (**את מצב הלוח לפני המהלך שהפסיד**)
* לוודא שמתים אחרי X צעדים בלי אוכל
* לוודא שהלבן מת קודם אם זה קורה
* לוודא שאכילת אוכל מאפס את ספירת הרעב
* לוודא שאפשר לצעוד לאיפה שהזנב היה בתור הקודם
* הרבה חוטים בהרבה משחקים: כתיבות בודדות. לוודא שיש ניהול תורות כמו שצריך
* הרבה חוטים בהרבה משחקים: כותבים מספר מהלכים במכה. כנ"ל.
* לוודא שכתיבת תו לא חוקי גורם להפסד
* לוודא שכתיבת תו לא חוקי בתו ה-N גורם להפסד בתור ה-N (ולא לפני)
* כמה חוטים (שהם אותו שחקן) מנסים לכתוב (בכל מיני צורות). לוודא שניהול התורות כמו שצריך
* לוודא שכתיבת 0 תווים מחזיר 0
* לוודא שכתיבת N תווים (מוצלחים) מחזיר N
* לוודא שכתיבת N+M תווים כשרק N מהם חוקיים מחזיר **שגיאה (פיאצה 392)**
* לוודא שכתיבה של N+M תווים כשהתו ה-N+1 גורם להפסד מחזיר **שגיאה (פיאצה 392)**
* לוודא שכתיבה במשחק גמור נכשלת (**מה הפלט?**)
* לוודא שכתיבה תוך כדי שהשחקן השני פורש לא נתקע (**ומה הפלט?**)
* לוודא שכאשר תהליך אחד עושה close בזמן שהשני כותב (אותו שחקן!) השחקן לא נתקע. לבדוק את זה עם הרבה שחקנים – לדוגמה, 2 תהליכים ממתינים לכתיבה (בתור השחקן הלבן) בזמן שהשחור יוצא מהמשחק.
* לנסות להיתקע בקירות הלוח ולראות שמפסידים
* לוודא שכתיבה עם NULL בבאפר מחזיר EFAULT (פיאצה 429)

ioctl:

* לבדוק את GET\_WINNER לפני שמישהו ניצח (**מה צריך להחזיר?**)
* לבדוק GET\_WINNER אחרי ניצחון (בכל צבע)
* לוודא שזה נכשל לחלוטין אחרי שמישהו עושה CLOSE
* לוודא שאפשר לקרוא כמה פעמים לפונקציה GET\_WINNER
* לוודא ש-GET\_COLOR עובד לפני ניצחון
* לוודא ש-GET\_COLOR עובד אחרי ניצחון
* לוודא ש-GET\_COLOR לא עובד אחרי שמישהו יוצא
* אם הפקודה לא קיימת, להחזיר ENOTTY