איך לקמפל קרנל (Linux)

מאת bindh3x

הקדמה

כולנו מריצים לינוקס. הראוטר שלכם, הטאבלט, הסמארטפון ואפילו השרת שממנו הורדתם את קובץ ה-PDF הזה. אני לא חושב שלפני 25 שנה לינוס טורבלדס, הבחור שהביא לנו את לינוקס, תיאר לעצמו ש-80% מהשרתים ברשת האינטרנט יריצו את הקרנל שהוא כתב.

וזה נכון. כי אם הייתם שואלים מישהו לפני כמה שנים אם הוא יהיה מוכן להשתמש "בהפצת לינוקס" כ-Desktop הוא כנראה היה צוחק לכם בפנים. האינטרקציה היחידה שהיתה לרוב האנשים עם לינוקס, היתה דרך שרת ה-FTP שלהם.

אבל עם השנים פותחו שולחנות עבודה בטעמים שונים, ההתעניינות הלכה וגדלה, ואיתה כמות המשתמשים "שהיגרו" מ-Windows.

אני אישית התרגלתי. כשאני רוצה לדעת איך פקודה מסוימת עובדת, קוד המקור שלה תמיד יהיה זמין. כשאני ארצה לדעת מה המבנה של חבילת ARP אני פשוט אציץ בקובץ ה-header המתאים. ואם אתם מתכנתים ב-C, אתם תרגישו בבית. אנחנו מקבלים גישה למערכת הפעלה ממש עד הברזלים.

כבר שנים שאני מקמפל את הקרנל שלי. כן, זה גוזל זמן, ודורש טיפה יותר ידע. אבל זה לא מדע טילים. יוצא לי להתקל בהרבה אנשים שמעדיפים להשתמש בקרנל שהחבר'ה בארץ' או דביאן (הפצות לינוקס) קימפלו עבורם "כי אין להם זמן או ידע להתעסק בזה" (לא שזה רע כן?).

במאמר הבא אני אציג את היתרונות והחסרונות בקימפול קרנל, מה זה patches, כיצד להתחיל ובאיזה להשתמש, ולבסוף גם כיצד לקמפל את הקרנל.

הטקסט שלפניכם נכתב בתחילה כפוסט בבלוג שלי (bindh3x.io), והורחב למאמר הנוכחי. אני לא מתיימר להיות "מומחה", ואני בטוח שחלק ממכם מקפלים את הקרנל בצורה שונה, או יותר יעילה. מטרת המאמר בסופו של דבר היא להציג את תהליך הקימפול עצמו.

למה לא לקמפל?

אם אתם משתמשים "רגילים" - כלומר: לא מפתחים מודולים לקרנל, המכונה שלכם לא מכילה חומרה שמצריכה קימפול קרנל, או שאתם מקמפלים כדי שהשם שלכם יופיע בבאנר של uname, אל תבזבזו את הזמן שלכם. תמשיכו להשתמש בקרנל שהמפתחים בהפצה שלכם קימפלו, הם עושים עבודה טובה, ונדיר שהמחשב שלכם לא יעבוד איתו (כן, גם אם הוא יוצר ב-2005).

למה כן לקמפל?

חשוב לזכור: קימפול קרנל הוא אינו תהליך - "בלתי הפיך". גם אם קימפלתם את הקרנל בעיניים עצומות, תמיד תוכלו להעלות את המערכת שלכם מהקרנל הקודם והכל יחזור לקדמותו.

1. יציבות

הפתרון הקל לקרנל יציב הוא להשתמש בגרסאות ה-LTS למינהם, שנמצאות במאגרים הרשמיים של הפצת הלינוקס שבה אתם משתמשים. בדרך כלל כל הפצה תחזיק גרסה יציבה (LTS), וגרסה עדכנית. אין בזה שום בעיה ולרוב הגרסאות הללו יהיו יציבות מאוד אך עלולות להכיל חורי אבטחה. מה שמביא אותי לדבר על הסיבה הבאה.

2. אבטחה

כאשר אתם מקפלים קרנל בעצמכם יש לכם שליטה מלאה בהגדרות האבטחה של הקרנל, תוכלו להתאים כל דבר שנראה לכם (בהנחה שאתם יודעים מה אתם עושים) מ-ASLR עד - SELinux ולהקשיח את המערכת שלכם.

3. אופטימיזציה

תוכלו לגרום למערכת שלכם לעבוד ביעילות, לבטל הגדרות שפוגעות בביצועי המערכת, למנוע קימפול של מודולים שאינם בשימוש ועוד. נרחיב על כך בהמשך.

מתחילים

הדבר הראשון שנצטרך הוא את קוד המקור של לינוקס. ניתן להוריד אותו מ-kernel.org. תבחרו גרסה שמתאימה לכם. הנוסחה היא פשוטה: אם הגרסה של הקרנל שרץ לכם כרגע במחשב יציבה תבחרו בה.

```
$ uname -a
Linux 4.12.12-1 #1 SMP PREEMPT Sun Sep 10 09:41:14 CEST 2017 x86_64
GNU/Linux
```

אני אבחר בגרסה: 4.11.1 - בשבילי היא יציבה מספיק. לאחר ההורדה צרו תיקייה בשם linux וחלצו לתוכה את קוד המקור של הקרנל.

```
$ mkdir linux
$ mv downloads/linux-4.11.1.tar.xz linux/
$ cd linux && tar xvf linux-4.11.1.tar.xz
```

לאחר שחילצנו את הקרנל, נעבור לחלק היותר מעניין: החלת patches - למי שלא יודע פאטץ' (patch) - הוא טלאי לקוד מסוים, בשימוש בטלאי ניתן להוסיף או להסיר קטעי קוד. הרבה kernel hackers כתבו patches שנועדו לשפר את ביצועי הקרנל, אך אינם חלק מה-mainline ולכן נצטרך להוריד אותם בנפרד.

Patches

ה-patch הראשון שבו נשתמש הוא של Con Kolivas - בחור אוסטרלי מוכר מאוד בסצנה שבין היתר כתב את LRZIP. בשנת 2009 Kolivas כתב את - Brain Fuck Scheduler. ה-Process scheduler כחלופה ל-CFS - אני לא ארחיב על Scheduling במאמר זה. בשורה התחתונה BFS - ישפר את ביצועי המערכת שלכם באופן משמעותי אם את משתמשים ב-Desktop. החל מלינוקס 4.8 - BFS כבר לא בפיתוח ונכתב מחדש כ-MUQss.

ניתן להוריד את ה-patch מכאן לפי גרסת הקרנל שאנחנו מקמפלים, במקרה שלי זה-4.11.1. לאחר ההורדה מקמו את ה-patch בתיקיית השורש של הקרנל שלכם:

```
$ pwd
~/linux
$ mv 4.11-sched-MuQSS_156.patch linux-4.11.1/
$ ls linux-4.11.1/
4.11-sched-MuQSS_156.patch  crypto         init     MAINTAINERS  scripts
arch                        Documentation  ipc      Makefile
security
block                       drivers        Kbuild   mm           sound
certs                       firmware       Kconfig  net          tools
COPYING                     fs             kernel   README       usr
CREDITS                     include        lib      samples      virt
```

הפאטץ' הבא שבו נשתמש הוא - Budget Fair Queueing ה-I/O scheduler. ניתן להוריד אותו מכאן. אם אתם משתמשים בקרנל 4.12.0 ומעלה אין צורך להוריד את BFQ, לינוקס כבר מכיל אותו.

לאחר שהורדתם את ה-patches נחיל אותם על קוד המקור של הקרנל. לפי הסדר:

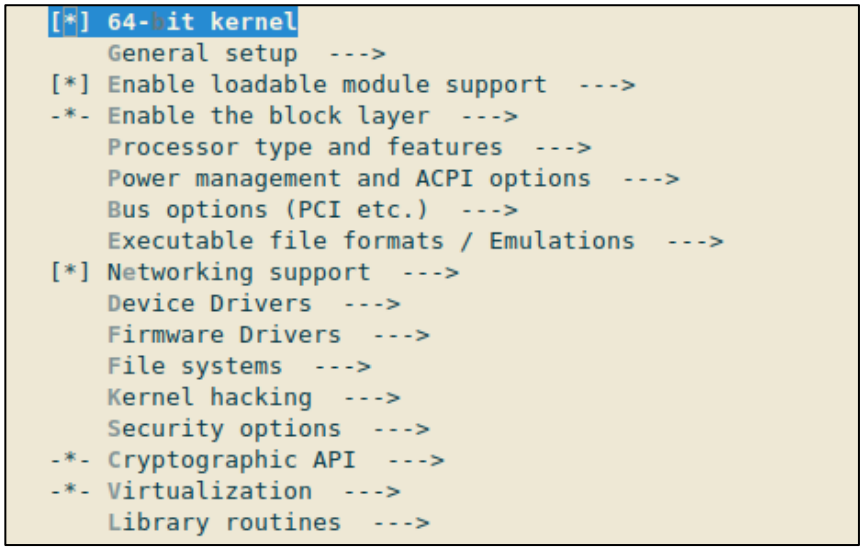
```
$ patch -p1 < 0001-block-cgroups-kconfig-build-bits-for-BFQ-v7r11-
4.11..patches
$ patch -p1 < 0002-block-introduce-the-BFQ-v7r11-I-O-sched-for-
4.11.0.patch
$ patch -p1 < 0003-block-bfq-add-Early-Queue-Merge-EQM-to-BFQ-v7r11-
for.patch
$ patch -p1 < 0004-blk-bfq-turn-BFQ-v7r11-for-4.11.0-into-BFQ-v8r11-
for.patch
$ patch -p1 < 4.11-sched-MuQSS_156.patch
```

ונעבור לחלק "הטריקי" - הגדרת הקרנל. לפני שתמשיכו וודאו שהחבילה bc מותקנת בהפצה שלכם.

הקרנל שרץ לכם כרגע במערכת מוגדר באמצעות קובץ config שנמצא במיקום /proc/config.gz, נשתמש בקובץ זה על מנת לשמור הגדרות קיימות בקרנל. הכנת הקרנל:

```
$ make mrproper
$ zcat /proc/config.gz > .config
$ make menuconfig
```

בצורה זאת אנחנו חוסכים לעצמנו זמן יקר, ומשתמשים בהגדרות שכבר עובדות לנו בקרנל, כך שמה שנשאר להגדיר הוא מינימלי. לאחר הרצת הפקודות תקבלו תפריט שנראה כך:

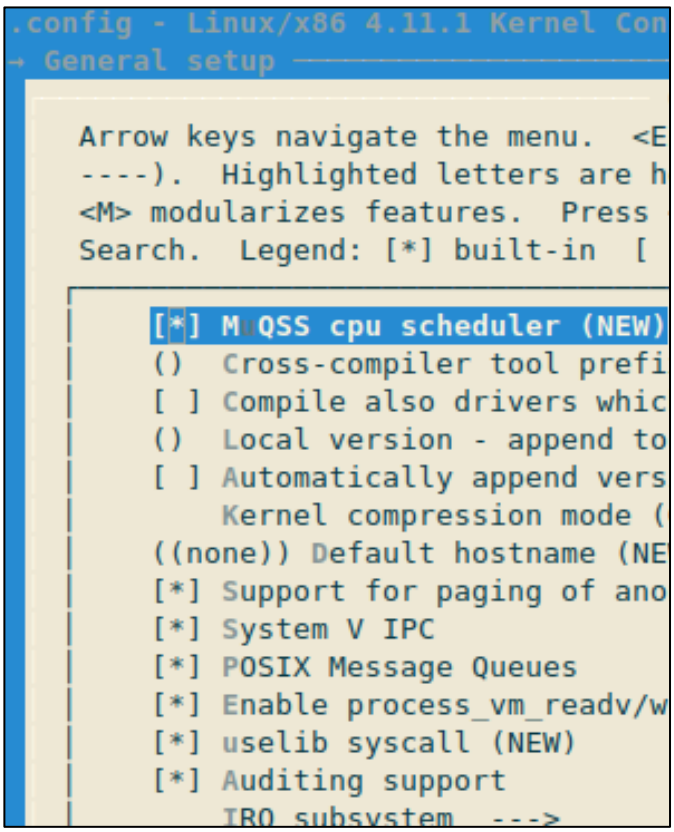


נסקור את האפשרויות השונות:

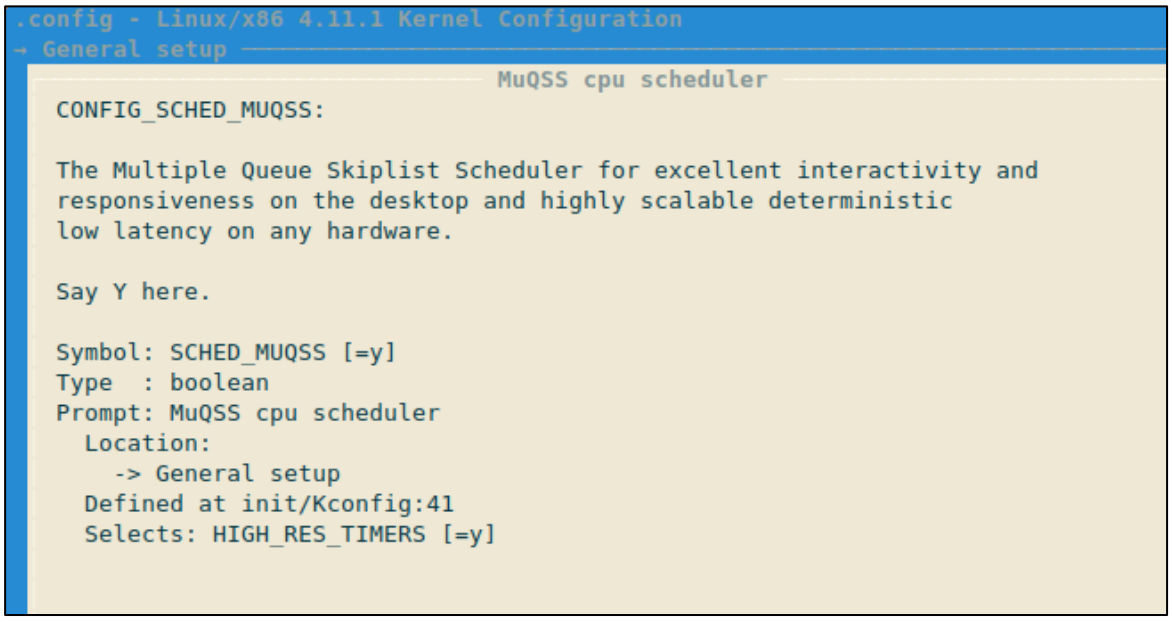
אפשרות	תיאור
General setup	הגדרות כלליות (משמש בין היתר להגדרת ה-CPU Scheduler).
Enable loadable module support	הגדרת תמיכה ב-LKM (מודולים נטענים).
Enable the block layer	תמיכה ב-block devices.
Processor type and features	כל מה שקשור למעבד כגון: Processor family, Timer frequency ועוד.
Power management and acpi options	ניהול צריכת חשמל, שינה, ACPI, Suspend-to-RAM.
BUS Options (PCI etc..)	תמיכה ב-PCI, PCI debugging וכו.
Executable file formats / Emulations	קבצי הרצה (ELF/MISC) וכ'ו.
Networking support	רשתות (sockets, netfilter, testing)
Device drivers.	Device drivers.
Firmware drivers	Firmware drivers.
File systems	תמיכה במערכות קבצים ext4/xfs/jfs (הצפנה, אבטחה וכ'ו)
Kernel hacking	הגדרות שונות של הקרנל, debugging,testing מיועד בעיקר למפתחים (או למשתמשים שרוצים לדעת קצת יותר).
Security options.	הגדרות אבטחה. (security module, בקרת גישה)
Cryptographic options	קריפטוגרפיה, אלגוריתמים (RSA/DH/SHA).
Virtualization	תמיכה בוירטואליזציה. (KVM וכ'ו.)
Library routines	Library routines.

החומרה שלי ככל הנראה שונה משלכם. תחפשו הגדרות מומלצות לחומרה שלכם, אתם לא רוצים למצוא את עצמכם עם קרנל שנכשל בתהליך ה-boot שזה עוד נחשב "למקרה טוב", במקרה הרע המערכת שלכם כן תריץ את הקרנל, ולא תבינו למה אתם מקבלים "Kernel Panic" אחרי רבע שעה. בכדי לישר קו נוודא שההגדרות התואמות ל-Patches שהחלנו מופעלות.

עברו למיקום General setup וודאו ש-MuQSS מסומן:



לקבלת מידע על אפשרות, תקישו - h:

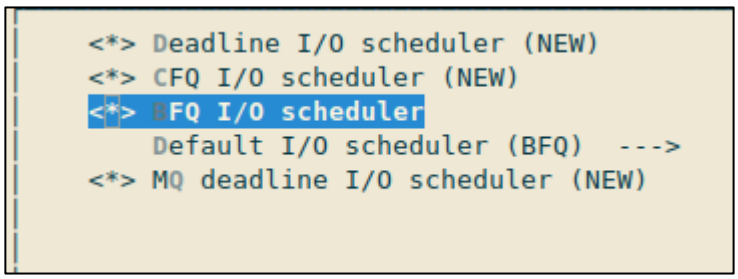


כפי שניתן לראות תפריט ה-"help" מסביר בפירוט על כל אפשרות, ונותן המלצה האם כדי להפעיל אותה או לא.

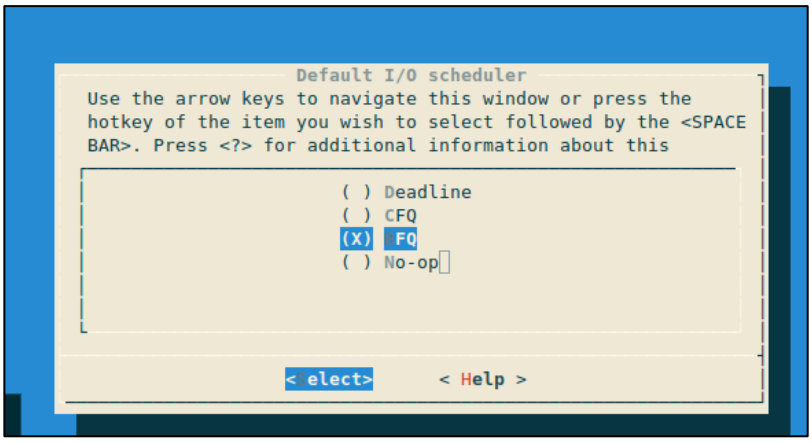
האפשרות השנייה שנגדיר היא ה-I/O Scheduler. עברו למיקום:

Enable the block layer -> IO Schedulers

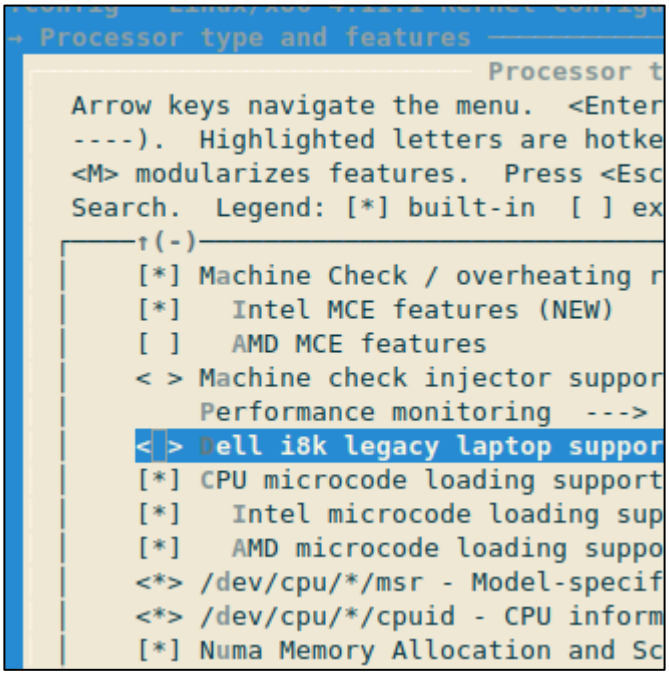
ותאפשרו את BFQ:



לאחר מכן עברו לאפשרות Default IO Scheduler ושנו ל-BFQ:



את הקרנל הנוכחי אני מקמפל ללפטופ HP עם מעבד של intel, לכן מודולים שמיועדים ל-Dell/IBM/AMD - לא רלוונטים עבורי.



תפעילו שיקול דעת, ומה שאתם לא מכירים - פשוט אל **תגדירו**.

אבטחה

בכדי להסביר את מודל האבטחה בלינוקס נחזור למקורות. לינוקס הוא נגזרת של - Unix, ולכן מודל האבטחה בברירת מחדל הוא DAC (Discretionary Access Control) - שבקצרה נותן לבעלים של קובץ\משאב\תהליך להחליט למי לתת להם גישה.

למשל: בוב יוצר קובץ בשם test.txt ורוצה לשתף אותו עם משמש אחר במערכת בשם ג'ון. על מנת לעשות זאת, בוב מריץ את הפקודה:

```
$ setfacl -m u:john:rwx text.txt
```

ומאפשר לג'ון גישה לאותו קובץ. מכאן אנחנו מבינים שבעל הקובץ שולט בהגדרות האבטחה שלו.
קיים מודל אבטחה נוסף בשם MAC (Mandatory Access Control) שמאפשר להרחיב את האבטחה מעבר למה שמודל DAC מציע לנו ולהקטין את הנזק בעת גישה לא מורשת למערכת.

Selinux

Security-Enhanced Linux - לינוקס עם אבטחה מתקדמת. הנושא עצמו מצריך מאמר נפרד. אך על קצה המזלג, SElinux הוא מימוש של MAC שעליו דיברנו מקודם. ההרחבה מוסיפה שכבת אבטחה נוספת מעל ה-DAC המסורתי, ומבצעת בקרת גישה באמצעות "Security policy".

Grsecurity/PaX

פאטץ' שנוצר על מנת לשפר את האבטחה בקרנל. ה-patch עצמו היה חופשי עד לשנת 2015, אז מפתחי ה-patch שיווקו את הגרסאות היציבות רק ללקוחות, ואיפשרו לציבור להשתמש בגרסאות ה-testing. מתחילת 2017 מפתחי grs כבר אינם מציעים הורדה של ה-patch בחינם. אם אתם מתכוונים לרכוש את ה-patch או ללמוד עליו קצת יותר, אתם מוזמנים לקרוא את המאמר "rsecurity Security Features for G the Linux Kernel" שנכתב על ידי גילי ינקוביץ' ופורסם בגליון ה-70 של המגזין.
כמובן שקיימים עוד פיצ'רים באבטחת לינוקס (רשתות, הצפנה, מודולים נוספים וכ'ו), אך זה אינו נושא המאמר.

קימפול

הגיע הזמן. אחרי שהכנו את הקרנל הגיע הזמן להכניס אותו לתנור. תריצו:

```
$ make -j4 # 4 cores = -j4
```

זה יקח קצת זמן. אז תתאזרו בסבלנות.תהליך הקימפול יכול לקחת בין 10~ / 40~ דקות, תלוי במעבד, כמות המודולים שאתם מקמפלים ועוד . מומלץ לסגור דפדפנים פתוחים או כל יישום שלוקח כח עיבוד.

כשהקרנל יהיה מוכן, תקבלו את ההודעה:

```
Kernel: arch/x86/boot/bzImage is ready  (#1)
```

לאחר הקימפול נתקין את המודולים החדשים:

```
$ sudo make modules_install
```

במידה ואתם מפתחים מודולים לקרנל אתם תצטרכו את ה-headers. תתקינו אותם עם הפקודה:

```
$ sudo make headers_install
```

נעתיק את הקרנל החדש לתקיית ה-/boot:

```
$ sudo cp -v arch/x86_64/boot/bzImage /boot/vmlinuz-linux-bh
```

ניצור initial RAM disk:

```
$ sudo mkinitcpio -k 4.11.1-bh -g /boot/initramfs-linux-bh
```

ולסיום נוסיף את הקרנל החדש לתפריט של - grub:

```
$ sudo grub-mkconfig -o /boot/grub/grub.cfg
```

זהו! כל מה שנשאר הוא לבצע reboot ולהנות מהקרנל החדש.

סיכום

יש יתרונות וחסרונות בקימפול הקרנל שלכם, מבחינתי היתרונות עולות על החסרונות. את קוד המקור של הקרנל שקימפלתם, אל תמחקו יכול להיות שתצטרכו אותו בעתיד. בכל תהליך הקימפול הנחתי שכבר יש לכם קרנל במערכת, קימפול על מערכת ללא קרנל מצריך יותר התעסקות.

במידה ואתם משתמשים ב-Archlinux תוכלו לקצר את כל התהליך הידני ולהשתמש ב-PKGBUILD. לדוגמא: linux-ck.