ساختار پارتیشین‌ها در MBR

ابوالفضل کاظمی (@0xAA55)

آیا تا به حال این سوال برایتان پیش آمده است که حداکثر تعداد پارتیشن بر روی یک هارد به چه تعداد است؟ یا اینکه بعد از روشن کردن کامپیوتر چه اتفاقی افتاده و از کجا مشخص می‌شود که از روی کدام پارتیشن سیستم‌عامل بارگذاری شده و شروع به اجرا شدن کند؟ آیا به این نکته توجه نموده‌اید که کوچکترین واحد ذخیره‌سازی اطلاعات بر روی دیسک چقدر بوده و آیا این مقدار برابر یک بایت بوده و یا مقدار بیشتری دارد؟ (شاید مقدار کوچک‌ترین واحد ذخیره‌سازی اطلاعات به نظر بی اهمیت برسد ولی در سرعت خواندن/نوشتن و میزان فضای هرز ایجاد شده بر روی دیسک تاثیر زیادی دارد) در این مقاله قصد داریم به این سوالات و سوالات دیگری پیرامون این موضوع پاسخ گوییم. در مقاله‌ی بعدی طریقه‌ی نوشتن یک برنامه به زبان پایتون برای استخراج اطلاعات پارتیشن‌ها، به کمک اطلاعات این مقاله را بیان خواهیم نمود.

داده‌های ذخیره شده بر روی هارددیسک در بلوک‌هایی به اندازه‌ی ۵۱۲بایت ذخیره می‌شوند که به این بلوک‌ها sector گفته می‌شود. این مقدار، کوچکترین میزان فضایی است که داده‌ای بر روی هارددیسک اشغال می‌کند. یعنی اگر شما حتی یک حرف یک بایتی مثل A را ذخیره کنید، مقدار فضایی که اشغال کرده‌اید برابر با ۵۱۲بایت می‌باشد. البته قضیه به اینجا ختم نمی‌شود و به دلیل اینکه شما به صورت مستقیم و با استفاده از interrupt داده بر روی دیسک ذخیره نکرده و از سیستم‌عامل (در واقع با استفاده از SystemCall، به صورت مستقیم یا غیر مستقیم) کمک گرفته و از لایه‌ی File System عبور کرده و سپس داده را بر روی دیسک و در یک **کلاستر** ذخیره می‌کنید، مقدار فضای اشغال شده از این نیز بیشتر شده و مضربی از ۵۱۲بایت خواهد بود. در بسیاری از سیستم‌های فایل امروزی مثل NTFS, Ext4 مقدار پیش‌فرض اندازه‌ی کلاستر برابر 4KB می‌باشد. (البته این مقدار برای پارتیشن‌ها با اندازه‌های مختلف می‌تواند متفاوت می‌باشد)

اولین sector هارددیسک (اولین 512bytes اشغال شده) که مهمترین سکتور نیز است، با عنوان Master Boot Record (MBR) شناخته می‌شود که پس از روشن شدن کامپیوتر و بارگذاری BIOS خوانده شده و اطلاعاتی دارد که برای پیدا کردن سیستم‌عامل، بارگذاری آن و دسترسی به اطلاعات پارتیشن‌ها مورد نیاز است و در صورتیکه این سکتور خراب شده و یا قابل دسترسی نباشد، با خطایی مواجه شده و امکان شروع به کار سیستم‌عامل وجود نخواهد داشت. در این ۵۱۲بایت مقدار ۶۴بایت برای ذخیره‌ی اطلاعات پارتیشن‌ها در نظر گرفته شده است که با استفاده از آن امکان تعریف ۴ پارتیشن وجود داشته که برای هر پارتیشن ۱۶بایت اطلاعات ذخیره می‌شود. پس با توضیحات ارائه شده مشخص می‌شود که بر روی هر دیسک ۴ پارتیشن بیشتر نمی‌توان ایجاد نمود. (مگر در حالت استفاده از درایو منطقی که در ادامه در مورد آن صحبت خواهد شد) از این ۴ پارتیشن **یکی** می‌تواند به عنوان پارتیشن Extended معرفی شده (و یا اصلا وجود نداشته باشد) و مابقی پارتیشن‌های Primary می‌باشند.

نکته: این ساختار در کامپیوترهایی وجود دارد که از BIOS و MBR استفاده می‌کنند و در کامپیوترهایی که UEFI و GPT دارند وجود ندارد. در مقاله‌ای دیگر به ساختار GPT خواهیم پرداخت.

در MBR کد مربوط به پیدا کردن سیستم‌عامل و بارگذاری BootLoader آن در ۴۴۶ بایت ابتدایی سکتور نوشته شده و از Offset شماره ۴۴۶ مقدار ۶۴ بایت اطلاعات پارتیشن‌ها را در بر می‌گیرد. دوبایت آخر MBR نیز (446+64==510) همیشه مقدار 0x55AA را در بر می‌گیرند. (در مقاله‌ای دیگر طریقه‌ی نوشتن برنامه اسمبلی برای MBR را توضیح خواهیم داد)

نکته: فاصله نسبت به ابتدای محلی را Offset می‌نامند. در مثال ما فاصله از ابتدای سکتور مدنظر است.

در شکل زیر داده‌ی یک MBR به صورت Hex نمایش داده شده است. دستور dd امکان کپی‌برداری از اطلاعات دیسک را فراهم می‌کند و همانطور که در شکل نمایش داده شده است مقدار ۵۱۲بایت از ابتدای دیسک اول کپی شده و در فایلی به نام mbr.img ذخیره می‌شود. (نام دیسک‌های sata, scsi در لینوکس با sd شروع شده و پس از آن، اولین دیسک sda، دومی sdb و... نام‌گذاری می‌شوند)

دستور xxd نیز برای hex dump استفاده می‌شود که بایت‌های یک فایل را در مبنای ۱۶ نمایش می‌دهد. در سمت چپ Offset هرخط، در وسط مقدار در مبنای ۱۶ و در سمت راست مقدار ASCII اطلاعات نمایش داده شده است. در اطلاعات نمایش داده شده عبارت GRUB مشاهده می‌شود که از آن می‌توان برداشت کرد که این MBR مربوط به سیستمی لینوکسی می‌باشد! در خط آخر (از Offset شماره 1f4==500 تا 1ff==511) و در مکان دو بایت آخر (یعنی در مکان‌های ۵۱۰ و ۵۱۱) مقدار 0x55AA مشاهده می‌شود! در صورت عدم وجود این مقدار در انتهای MBR خطای Missing Operating System داده خواهد شد.

```
root@kalivm01:~# dd if=/dev/sda of=mbr.img bs=512 count=1
1+0 records in
1+0 records out
512 bytes copied, 0.000373863 s, 1.4 MB/s
root@kalivm01:~# xxd -c 25 mbr.img
00000000: eb63 9010 8ed0 bc00 b0b8 0000 8ed8 8ec0 fbbe 007c bf00 06b9 00  .c.................|.....
00000019: 02f3 a4ea 2106 0000 bebe 0738 0475 0b83 c610 81fe fe07 75f3 eb  ....!......8.u........u..
00000032: 16b4 02b0 01bb 007c b280 8a74 018b 4c02 cd13 ea00 7c00 00eb fe  .......|...t..L.....|....
0000004b: 0000 0000 0000 0000 0000 0000 0000 0000 8001 0000 0000 0000 00  .........................
00000064: fffa 9090 f6c2 8074 05f6 c270 7402 b280 ea79 7c00 0031 c08e d8  .......t...pt....y|..1...
0000007d: 8ed0 bc00 20fb a064 7c3c ff74 0288 c252 bb17 04f6 0703 7406 be  .... ..d|<.t...R......t..
00000096: 887d e817 01be 057c b441 bbaa 55cd 135a 5272 3d81 fb55 aa75 37  .}.....|.A..U..ZRr=..U.u7
000000af: 83e1 0174 3231 c089 4404 4088 44ff 8944 02c7 0410 0066 8b1e 5c  ...t21..D.@.D..D.....f..\
000000c8: 7c66 895c 0866 8b1e 607c 6689 5c0c c744 0600 70b4 42cd 1372 05  |f.\.f..`|f.\..D..p.B..r.
000000e1: bb00 70eb 76b4 08cd 1373 0d5a 84d2 0f83 d000 be93 7de9 8200 66  ..p.v....s.Z........}...f
000000fa: 0fb6 c688 64ff 4066 8944 040f b6d1 c1e2 0288 e888 f440 8944 08  ....d.@f.D...........@.D.
00000113: 0fb6 c2c0 e802 6689 0466 a160 7c66 09c0 754e 66a1 5c7c 6631 d2  ......f..f.`|f..uNf.\|f1.
0000012c: 66f7 3488 d131 d266 f774 043b 4408 7d37 fec1 88c5 30c0 c1e8 02  f.4..1.f.t.;D.}7....0....
00000145: 08c1 88d0 5a88 c6bb 0070 8ec3 31db b801 02cd 1372 1e8c c360 1e  ....Z....p..1......r...`.
0000015e: b900 018e db31 f6bf 0080 8ec6 fcf3 a51f 61ff 265a 7cbe 8e7d eb  .....1..........a.&Z|..}.
00000177: 03be 9d7d e834 00be a27d e82e 00cd 18eb fe47 5255 4220 0047 65  ...}.4...}.......GRUB .Ge
00000190: 6f6d 0048 6172 6420 4469 736b 0052 6561 6400 2045 7272 6f72 0d  om.Hard Disk.Read. Error.
000001a9: 0a00 bb01 00b4 0ecd 10ac 3c00 75f4 c322 0754 dc00 0080 2021 00  ..........<.u..".T.... !.
000001c2: 83fe ffff 0008 0000 0000 8018 00fe ffff 05fe ffff fe0f 8018 02  ........................
000001db: e87f 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00  ........................
000001f4: 0000 0000 0000 0000 0000 55aa                                    ..........U.
```

پس از مشاهده کردن محتویات یک MBR باز گردیم به بررسی ساختار پارتیشن‌ها در آن و اطلاعاتی که در ۱۶بایت مربوط به هر پارتیشن ذخیره می‌شود. از جمله اطلاعاتی که در ۱۶بایت مربوط به هر پارتیشن قرار می‌گیرد و برای ما اهمیت دارد موارد زیر می‌باشند:

- بایت اول (شماره‌ی صفر) Active بودن پارتیشن را مشخص کرده و می‌تواند دو مقدار 0x80 و 0 را داشته باشد. مقدار **0x80** که تنها به پارتیشن‌های Primary داده می‌شود، بیانگر پارتیشنی است که سیستم‌عامل بر روی آن نصب شده و باید Boot Loader سیستم‌عامل از روی آن بارگذاری شود. این مقدار تنها برای یک پارتیشن باید ست شده و در غیر این صورت در زمان بارگذاری سیستم‌عامل با خطای Invalid Boot Partition مواجه می‌شویم.
- بایت شماره‌ی ۴ نوع پارتیشن را مشخص می‌کند. به عنوان مثال مقدار 0 برای خالی بودن، 0x05 برای Extended، 0x07 برای NTFS و 0x83 برای Linux مورد استفاده قرار می‌گیرند.
- بایت‌های شماره‌ی ۸ تا ۱۱ برای Logical Block Addressing (LBA) مورد استفاده قرار می‌گیرند. (چهاربایت که اولین بایت، کمترین ارزش و آخرین بایت، بیشترین ارزش را دارد) از این مقدار برای Offset شروع پارتیشن استفاده می‌شود و تعداد **سکتوری** (دقت شود که سکتور و نه بایت) را مشخص می‌کند که از ابتدای دیسک باید رد کنیم تا به ابتدای پارتیشن برسیم.
- بایت‌های شماره‌ی ۱۲ تا ۱۵ نیز اندازه‌ی پارتیشن به تعداد سکتور را مشخص می‌کنند.

در شکل زیر بخش‌های توضیح داده شده و مابقی بخش‌های تشکیل دهنده‌ی MBR نمایش داده شده است.

DOS partition table format	
Bytes	Purpose
0-445	Boot code
446-461	Partition Table Entry #1
462-477	Partition Table Entry #2
478-493	Partition Table Entry #3
494-509	Partition Table Entry #4
510-511	Signature value (0xAA55)

DOS Partition Table Entry format	
Bytes	Purpose
0	Bootable flag (0x80=active; else 0x00)
1-3	Starting CHS address
4	Partition type (e.g., 0x00=empty, 0x01=FAT12, 0x07=NTFS, 0x0b=FAT32 (CHS), 0x83=Linux, 0xa5=FreeBSD, 0xa8=MacOS X)*
5-7	Ending CHS address
8-11	Starting LBA address
12-15	Size (in sectors)

نکته: در این شکل برای Signature مقدار 0xAA55 ذکر شده است در حالیکه با دیدن Hex مربوط به MBR ، عبارت 0x55AA در خروجی وجود داشت که یکی نمایش Big-Endian و دیگری Little-Endian است.

برای درک دقیق‌تر موضوع، مقدار LBA و اندازه‌ی پارتیشن اول MBR نمایش داده شده را محاسبه کرده و با خروجی دستور fdisk مقایسه می‌کنیم. شروع اولین پارتیشن از Offset با مقدار 446 می‌باشد که برای مقدار LBA باید بایت‌های ۸ تا ۱۱ را برداشته و مقدار آنرا با بدست آوردن یک عدد ۳۲ بیتی محاسبه نماییم. پس باید از آدرس 446+8==1C6 مقدار ۴بایت را برداریم. در شکل MBR خط سوم از پایین از آدرس 1C2 شروع می‌شود که با اضافه کردن ۴بایت به آن به آدرس 1C6 در ابتدای ستون سوم خواهیم رسید. پس ۳۲ بیتی که مقدار LBA را مشخص می‌کنند به صورت زیر بوده که با تبدیل به مبنای ۱۰ به عدد ۲۰۴۸ خواهیم رسید:

$$0x00080000 = 0x00 + 0x08 * 256 + 0x00 * 256^2 + 0x00 * 256^3 = 2048$$

چون ۳۲ بیت این عدد را به بخش‌های ۸ بیتی تقسیم نمودیم، (هر بایت برابر ۸ بیت می‌باشد) پس ارزش هربایت $2^8==256$ برابر ارزش بایت قبلی می‌باشد.

به طریق مشابه برای بدست آوردن اندازه‌ی پارتیشن باید به ابتدای بایت 446+12==1CA رفته و ۴بایت را برداشته و اندازه‌ی پارتیشن را بدست آورد. (۴ بایت خط سوم از پایین، ستون‌های ۵و۶)

$$0x00008018 = 0x00 + 0x00 * 256 + 0x80 * 256^2 + 0x18 * 256^3 = 411041792$$

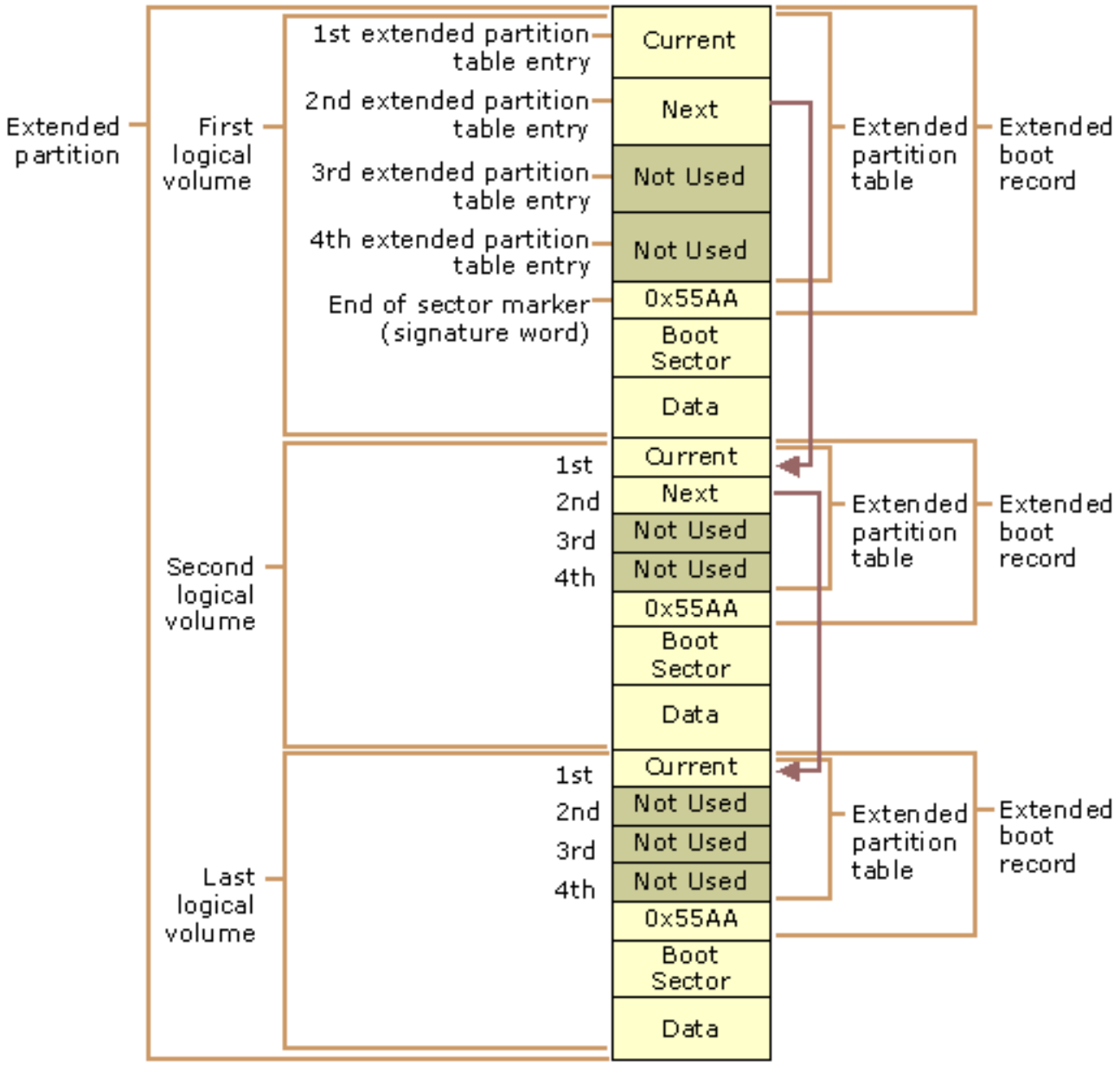
با اجرای دستور fdisk بر روی فایل mbr.img صحت این اطلاعات مشخص می‌شود. دقت شود که به دلیل کپی نگرفتن از تمامی اطلاعات دیسک، اطلاعات پارتیشن‌های Extended کامل نبوده و fdisk خطایی می‌دهد. (در ادامه با ساختار کامل این پارتیشن‌ها و دلیل این خطا آشنا می‌شویم)

```
root@kalivm01:~# fdisk -l mbr.img
Failed to read extended partition table (offset=411045886): No such file or directory
Disk mbr.img: 512 B, 512 bytes, 1 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xdc540722

Device     Boot     Start       End   Sectors  Size Id Type
mbr.img1   *         2048 411043839 411041792  196G 83 Linux
mbr.img2       411045886 419428351   8382466    4G  5 Extended
```

تا اینجای بحث توضیح دادیم که ۴ بخش برای اطلاعات ۴ پارتیشن وجود دارد که یکی از این ۴تا می‌تواند از نوع Extended باشد. این نوع پارتیشن امکان ایجاد Logical Drive (از دید سیستم‌عامل و کاربر نهایی همان پارتیشن و افزودن تعداد درایوها و رسیدن به تعداد درایور بیشتر از ۴) را فراهم می‌کند. روال کار به این شکل است که یک لیست پیوندی ایجاد شده که این پارتیشن Extended به عنوان Head عمل کرده و اشاره کننده به اولین Logical Drive است. (مقدار LBA این پارتیشن به ابتدای اولین سکتور از اولین درایو اشاره کرده و Size اندازه‌ی کلیه‌ی درایوها در مجموع و فضایی که تشکیل می‌دهند را مشخص می‌کند)

در هر Logical Drive اولین سکتور با نام Extended Boot Record (EBR) شناخته شده و ساختاری مشابه MBR دارد که از ۴ بخش مربوط به پارتیشن‌ها، تنها ۲ بخش استفاده شده که بخش مربوط به پارتیشن اول، اطلاعات درایو فعلی را در بر گرفته و بخش دوم اشاره‌گر به Logical Drive بعدی می‌باشد. در انتهای کار و پس از رسیدن به آخرین درایو، مقدار فیلد type (نوع پارتیشن در بایت شماره‌ی ۴) پارتیشن دوم EBR (اشاره‌گر به درایو بعدی) برابر صفر خواهد بود. (همان NULL در انتهای لیست پیوندی ساده) این ساختار در شکل زیر نمایش داده شده است.



نکته مهم: در Logical Driverها مقدار LBA مشخص شده در پارتیشن اول EBR، نسبت به درایو قبلی بوده و از ابتدای محل Extended نیست.

برای بررسی عملی توضیحات داده شده، با استفاده از دستور dd سکتور مربوط به EBR را خوانده و اطلاعات نمایش داده شده توسط آنرا بررسی می‌نماییم. اولین مرحله پیدا کردن آدرس EBR است. برای اینکار از شکل خروجی دستور fdisk -l mbr.img و در خط دوم که نوع پارتیشن Extended ذکر شده است، مقدار start را برداشته و این تعداد سکتور skip کرده و سپس 512 بایت می‌خوانیم. سپس دستور fdisk را برای فایل جدید ایجاد شده، اجرا کرده و LBA, Size درایو را بدست می‌آوریم.

```
root@kalivm01:~# dd count=1 bs=512 skip=411045886 if=/dev/sda of=ebr.img
1+0 records in
1+0 records out
512 bytes copied, 0.00042338 s, 1.2 MB/s
root@kalivm01:~# fdisk -l ebr.img
Disk ebr.img: 512 B, 512 bytes, 1 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device     Boot Start     End Sectors Size Id Type
ebr.img1           2 8382465 8382464   4G 82 Linux swap / Solaris
```

در این شکل مشخص است که درایومنطقی که درون Extended ایجاد شده است از نوع Swap بوده و شروع آن ۲سکتور بعد از شروع Extended می‌باشد (یعنی از سکتور 411045888 که برابر است با (skip+start(ebr.img

جمع بندی نهایی و نمایش کلیه‌ی اطلاعات دیسک با استفاده از اجرای fdisk بر روی sda به صورت زیر است:

```
root@kalivm01:~# fdisk -l /dev/sda
Disk /dev/sda: 200 GiB, 214748364800 bytes, 419430400 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xdc540722

Device     Boot     Start       End   Sectors  Size Id Type
/dev/sda1  *         2048 411043839 411041792  196G 83 Linux
/dev/sda2       411045886 419428351   8382466    4G  5 Extended
/dev/sda5       411045888 419428351   8382464    4G 82 Linux swap / Solaris
```

امیدوارم مفید بوده باشه، موفق باشید.