قطعه بش اسکریپتی که از توی خط‌فرمان صفحه نمایش رو کپچر میکنه به همراه تمام صداها

در گشت و گزارام به این تکه برنامه برخوردم که  وقتی امتحانش کردم متعجب شدم از خروجی که بهم داد ! بدون هیچ مشکلی کل صفحه رو همراه با صدای میکروفون و حتی آهنگی که داشتم با کامپیوتر پخش میکردم کپچر کرد!
قطعه بش اسکریپ ذکر شده به صورت زیر است:

```
ffmpeg -f alsa -itsoffset 00:00:02.000 -ac 2 -i hw:0,0 -f x11grab -s $(xwininfo
-root | grep 'geometry' | awk '{print $2;}') -r 10 -i :0.0 -sameq -f mp4 -s wvga
-y intro.mp4
```

بهترین راه برای یادگرفتن دستورات پیچیده و ترکیبی، اینه که اول اونها رو به ساده ترین شکل بنویسیم، بعد رفته رفته با مطالعه مستندات دستورها، سویچهای پیچیده تر رو اضافه کنیم تا در نهایت به همون شکل تر کیبی اولیه برسیم.
مستنداتی که در باره ffmpeg ازشون استفاده کردم، این ها هستند :

http://www.ffmpeg.org/ffmpeg-doc.html
http://www.ffmpeg.org/general.html
در مورد اینکه اصلا ffmpeg چیه، تو document اینطوری نوشته شده :

```
FFmpeg is a very fast video and audio converter. It can also grab from a live
audio/video source
```

پس ffmpeg ابزاریه هم برای تبدیل انواع فایلهای صوتی و تصویری، هم برای ذخیره کردن فایلهای صوتی و تصویری از منابع Live .

حالا یه نگاهی به syntax این دستور میندازم:

```
ffmpeg [[infile options][`-i' infile]]... {[outfile options] outfile...
```

تا همینجا معلومه که حداقل یک ورودی و یک خروجی برای ffmpeg باید مشخص بشه(infile, outfile) و یک توضیح بسیار مهم در مورد syntax تو همین document هست به این شکل :

```
As a general rule, options are applied to the next specified file. Therefore,
order is important, and you can have the same option on the command line multiple
times
```

پس هر option روی فایل بعد از خودش تاثیر میگذاره، می تونیم از یک option چند بار استفاده کنیم و ترتیب نوشته شدن option ها هم مهمه. ساده ترین شکل این دستور که میتونه خروجی مانیتور رو داخل یک فایل ویدیویی بدون صدا ذخیره کنه، اینه :

```
ffmpeg -f x11grab -s 1024x768 -r 10 -i :0.0 intro.mp4
```

با سویچ f- نوع ورودی مشخص شده، x11grab یعنی ورودی از طریق X11

با سویچ s- سایز ورودی مشخص شده
با سویچ r- سرعت فریم ها مشخص شده، 10 فریم در ثانیه
با سویچ i- نام ابزار ورودی مشخص شده، 0.0: یعنی اولین screen در اولین display ، در واقع تو این مثال میشه مانیتور
و در نهایت اسم فایل خروجی مشخص شده.
خوب حالا با خوندن document ها سعی میکنم از بقیه command سردر بیارم.
سویچ y- باعث میشه فایل جدید رو قبلی overwrite بشه.
سویچهای f- و s- همونطور که برای ورودی استفاده میشن برای خروجی هم قابل استفاده هستن
با سویچ sameq- میتونیم کیفیت تصویر ورودی و خروج رو یکسان نگهداریم.

پس تا اینجا میتونیم این شکل از دستور ffmpeg رو توضیح بدیم :

```
ffmpeg \
 -f x11grab -s 1024x768 -r 10 -i :0.0 -sameq \
 -f mp4 -s 1024x768 -y intro.mp4
```

تنها نکته ای که باید بگم اینه که برای خوانا شدن دستور با کمک علامت \ سویچهای مربوط به ورودی و خروجی رو در دو خط جدا از هم نوشتم، و اگر خواستید از این دستور استفاده کنید، دقت کنید که بعد از \ بلافاصله Enter بزنید، چون زدن هر کلیدی غیر از Enter مثلا space بعد از \، باعث میشه bash در تفسیر دستور اشتباه کنه.

نوبت می رسه به ذخیره صدا :
با سویچ f- میتونیم نوع ورودی صدا رو تنظیم کنیم، در این مثال از ALSA استفاده شده.
با سویچ ac- تعداد کانالهای ورودی صدا مشخص میشه
وبا سویچ i- سخت افزار ورودی صدا رو معرفی میکنیم.
تا الان دستور به این شکل تبدیل شده :

```
ffmpeg \
 -f alsa -ac 2 -i hw:0,0 \
 -f x11grab -s 1024x768 -r 10 -i :0.0 -sameq \
 -f mp4 -s 1024x768 -y intro.mp4
```

خط اول ورودی صدا، خط دوم ورودی تصویر و خط سوم خروجی فایل رو مشخص میکنه. البته اگه میخواید که ورودی صدا از طریق میکروفون انجام بشه ممکنه لازم باشه یه مقداری تنظیمات alsamixer رو تغییر بدین و اگه از تغییرات راضی بودین میتونید با دستور alsactl store ذخیره کنیدشون.
تنها سویچ itsoffset 00:00:02.000- به نظر بدرد نخور میاد، چون با حذف کردنش یا تغییر مقدار عدد 2 هیچ تغییری رو در خوروجی ندیدم، البته همونطور که تو document توضیح داده شده، این سویچ باعث میشه که چند ثانیه اول فایل خالی بمونه، و در مثال ما، از ثانیه 2 به بعد ویدیو ذخیره بشه، ولی اینطور که به نظر میرسه این سویچ فقط وقتی کار میکنه که تنها صدا یا تنها تصویر ذخیره بشه، و وقتی که صدا و تصویر همزمان ذخیره میشن هیچ تاثیر مشخصی روی خروجی نداره.
مشکل این دستور اینه که هر کسی خواست ازش استفاده کنه باید رزولیشن مانیتور خودش رو تو دستور بجای 1024x768 بنویسه، چون این روزها این رزولیشن فقط به درد مانیتور قدیمی من میخوره. به جاش میتونیم از خروجی دستور xwininfo طوری در ffmpeg استفاده کنیم که مجبور نباشیم روی سیستم های متفاوت دستور رو تغییر بدیم. اگه به خروجی دستور xwininfo -root توجه کنید میبینید که آخرین خط که با geometry- شروع میشه رزولیشن مانیتور رو نمایش میده :

```
root@challenger:~/capture # xwininfo -root

xwininfo: Window id: 0x40 (the root window) (has no name)
 Absolute upper-left X: 0
 Absolute upper-left Y: 0
 Relative upper-left X: 0
 Relative upper-left Y: 0
 Width: 1024
 Height: 768
 Depth: 24
 Visual Class: TrueColor
 Border width: 0
 Class: InputOutput
 Colormap: 0x20 (installed)
 Bit Gravity State: NorthWestGravity
 Window Gravity State: NorthWestGravity
 Backing Store State: NotUseful
 Save Under State: no
 Map State: IsViewable
 Override Redirect State: no
 Corners: +0+0 -0+0 -0-0 +0-0
 -geometry 1024x768+0+0
```

اول اون یک خط رو از خروجی جدا میکنم :

```
root@challenger:~/capture # xwininfo -root | grep geometry
 -geometry 1024x768+0+0
```

حالا از این یک خط، ستون دوم رو که شامل رزولیشن میشه، جدا میکنم :

```
root@challenger:~/capture # xwininfo -root | grep geometry | awk '{print$2}'
1024x768+0+0
```

و در نهایت با کمک $() خروجی این دستور رو در برابر سویچهای s- در دستور ffmpeg استفاده میکنم :

```
ffmpeg \
 -f alsa -ac 2 -i hw:0,0 \
 -f x11grab -s $(xwininfo -root | grep geometry | awk '{print$2}') \
 -r 10 -i :0.0 -sameq \
 -f mp4 -s $(xwininfo -root | grep geometry | awk '{print$2}') -y intro.mp4
```

نکته اینکه در دستوری که شما مثال زدید، پارامتر wvga به سویچ s- دوم پاس شده که باعث میشه همیشه خروجی با رزولیشن 852x480 تولید بشه، که به نظر من خیلی جالب نبود، و چون ترجیح میدم ورودی و خروجی از یک رزولیشن استفاده کنن، پارامترهای هر دو سویچ s- رو یکسان دادم.