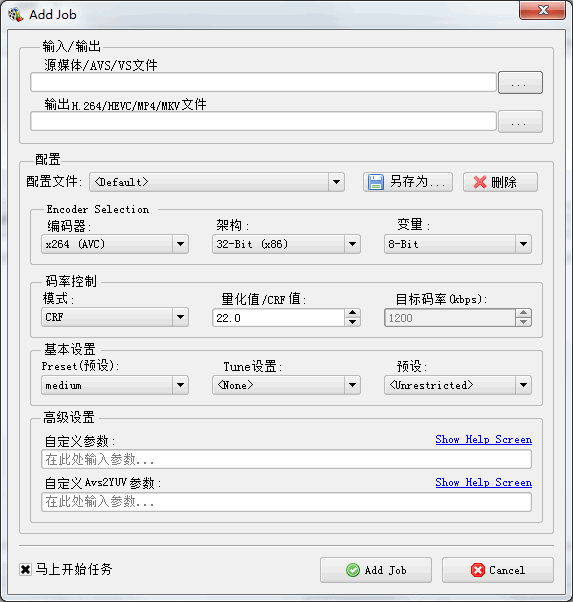
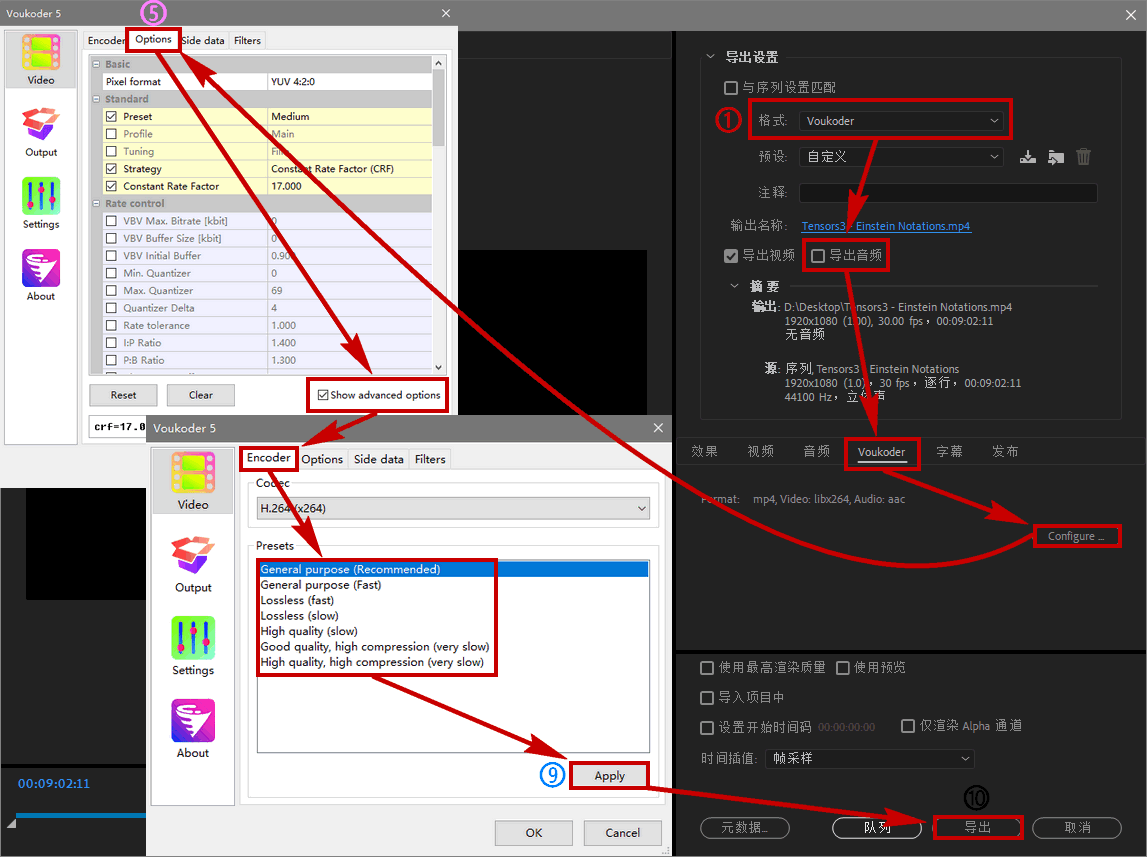
从x264教程综合版总结出的参数配置. 目的是把参数直接贴到软件里用, 本教程提供的参数不完美, 应根据自己情况修改

软件下载

[**小丸工具箱**](https://maruko.appinn.me/)

中文免费软件, 将视频拖拽入视频栏/批量压制栏中以导入视频, 点击自定义, 将参数粘贴进去, 选x264位深, 选输出格式就可以压制了

[**Simple x264 Launcher**](https://bitbucket.org/muldersoft/simple-x264-launcher/downloads)

英语, 小体积软件, 点Add new job进入Add Job页面, 将要压缩的视频, AVS或VS文件拖进输入框, 写输出名/后缀, 选择编码器, 架构, 变量, 模式, 量化值, preset, tune和profile, 把写好的参数复制粘贴到自定义参数栏(crf用上面的选择)就好了, 点击配置文件的另存为能保存这次的压制设置. 点Add Job开始压制

[**Voukoder & Connector**](https://www.voukoder.org/)

PR/VEGAS/AE/达芬奇导出插件

[**OBS直播/录屏**](https://obsproject.com/download)

支持MacOS, 设置复杂, 直播上是同类软件最强. 详细设置在本教程后半部分

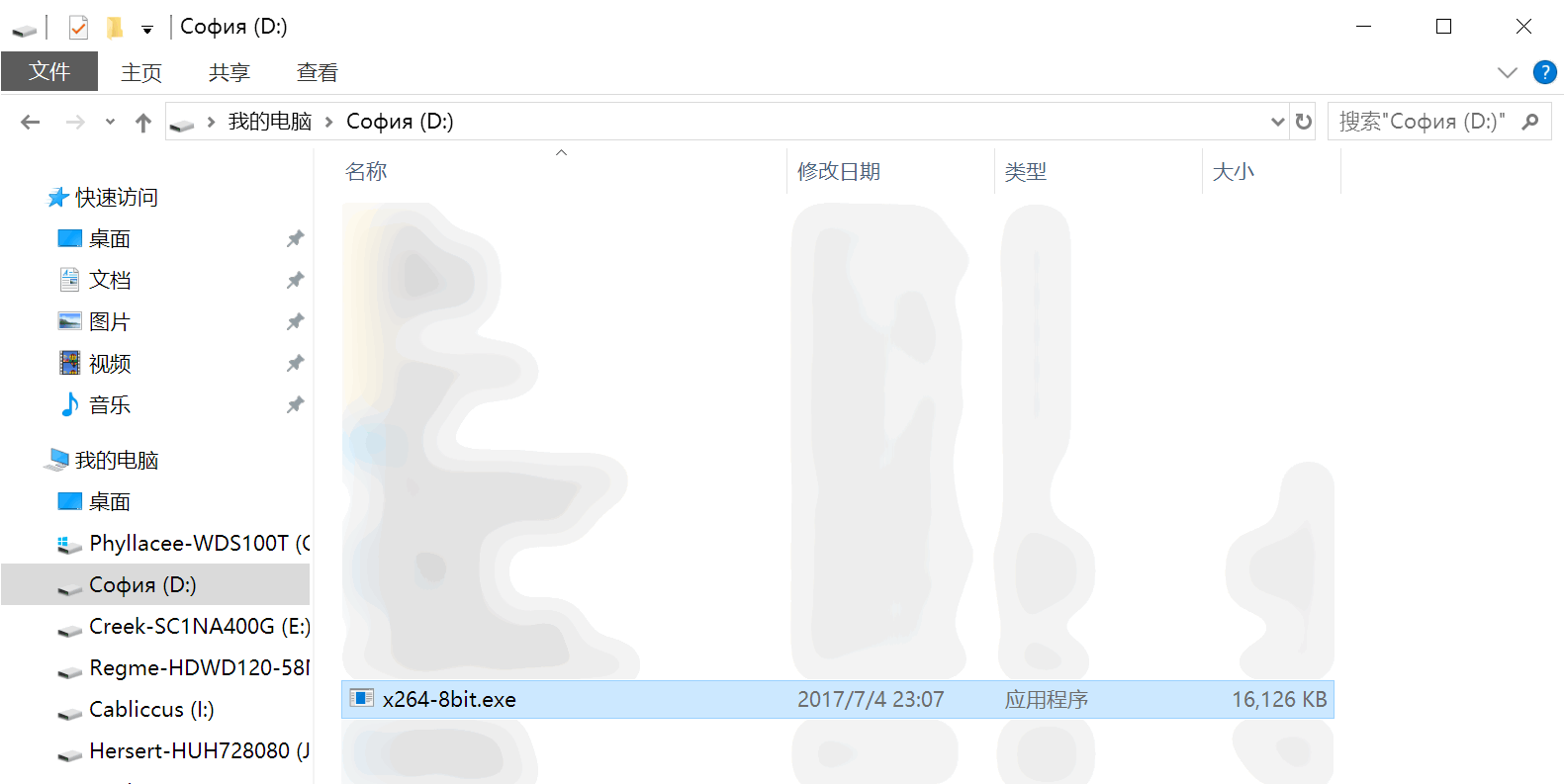
|  |
| --- |
| [**Patman x264**](https://www.mediafire.com/folder/arv5xmdqyiczc)√lavf编解码 √版本最新 |
| **7mod x264**[谷歌盘](https://drive.google.com/drive/folders/1kFCeNGA_wiiLt-DSeI3cyY8vxlffgQcy?usp=sharing)/[百度云](https://pan.baidu.com/s/1sbz8WztGTz3lcLzirHW_2w) √lavf编解码, √subme11, √hqdn3d降噪 |
| [**ffmpeg**](http://ffmpeg.org/download.html)**(全系统)**: 备用地址 ottverse.com/ffmpeg-builds |

### 压制软件工作流程图解（不严谨）

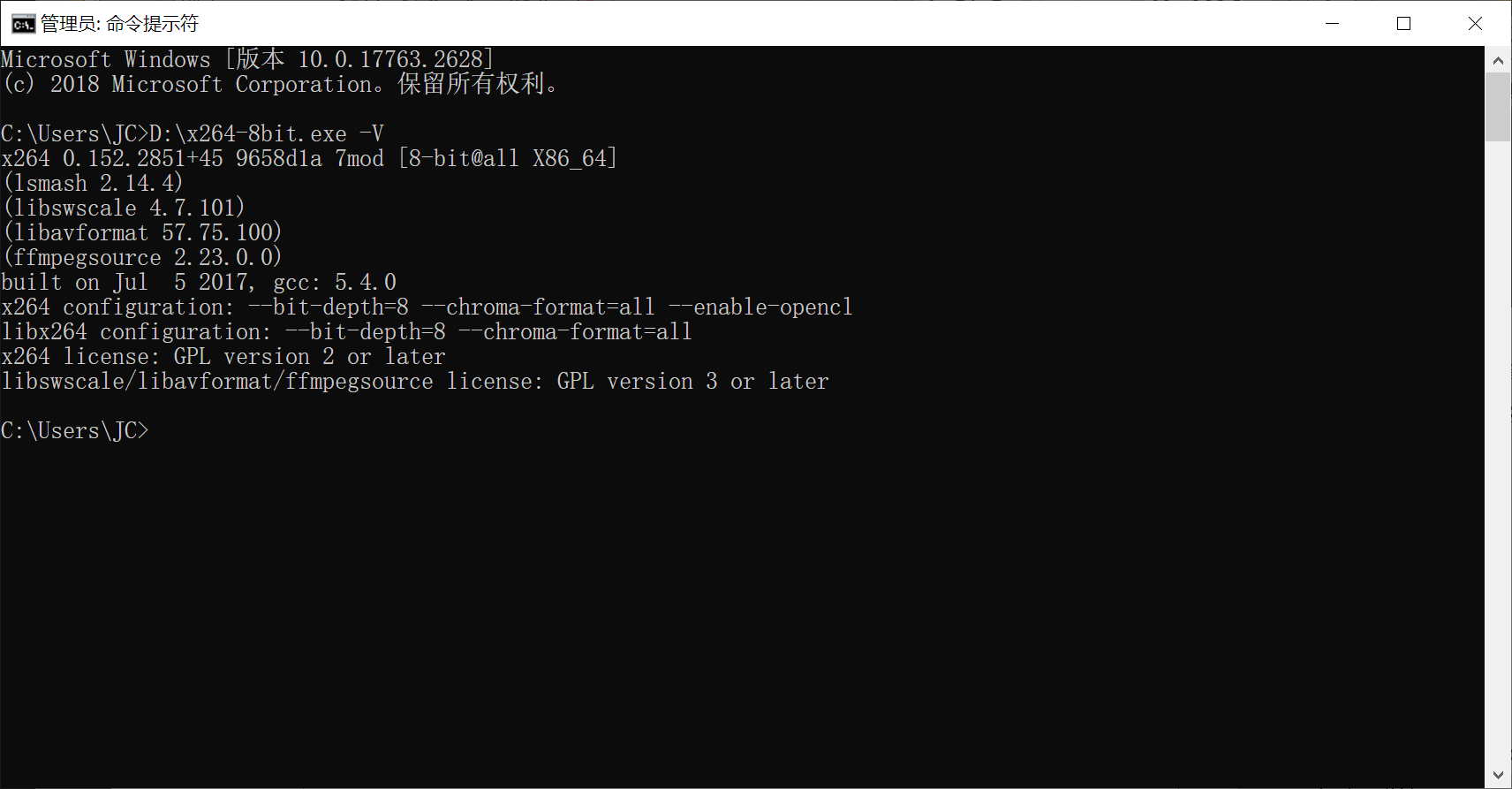


### x264.exe命令行参数用法教程(新人必看)

[照上表下载x264.exe并记住路径, 此处置于D盘根目录下]



[打开Windows自带的CMD.exe]



[CMD路径自动填充] 路径名写一半, 然后按[Tab]直到文件名匹配为止

[引用程序] C:\文件夹\x264.exe

[CLI参数] --me esa --merange 48 --keyint 200 […]

[x264/5的导出命令和导入] --output C:\文件夹\导出.mp4 C:\文件夹\导入.mp4

[参数格式] x264.exe --me esa --merange 24 […] --output "导出.mp4" "导入.mp4"

[参数用例] D:\x264-8bit.exe --me umh --subme 11 --merange 32 -I 270 -i 1 -b 11 --b-adapt 2 -r 3 --direct auto --crf 19 --qpmin 13 --rc-lookahead 90 --aq-mode 3 --aq-strength 1 --trellis 2 --deblock 0:0 --psy-rd 0.7:0.2 --fullrange --vf hqdn3d:1.1,1.1,1.1,1.1 --output "F:\导出.mp4" "D:\导入.mp4"

### CMD/Bash报错自动导出:

x265.exe [参数] 2>C:\x265报错.txt

### 三角形定律

见右图

### QAAC压制音频见[教程](https://www.nazorip.site/archives/44/)或[Github](https://github.com/iAvoe/QAAC-Tutorial-Standalone/blob/master/%E6%95%99%E7%A8%8B.md)

### ffmpeg .ass字幕渲染滤镜:

-filter\_complex "ass='F\:/字幕.ass'"

### 中途停止压制, 并封装现有帧为视频:

Ctrl+C, 部分人编译的x265.exe自带功能

### QAAC 压制音频见[教程](https://www.nazorip.site/archives/44/)或[Github](https://github.com/iAvoe/QAAC-Tutorial-Standalone/blob/master/%E6%95%99%E7%A8%8B.md)

### ffmpeg内置缩放:

例: -sws\_flags bitexact+full\_chroma\_int+full\_chroma\_inp+accurate\_rnd

* -sws\_flags bicubic/bitexact/gauss/bicublin/lanczos/spline/+full\_chroma\_int/+full\_chroma\_inp/+accurate\_rnd

### ffmpeg 封装视音频, 更改导出文件后缀名以指定封装格式(.mkv格式还支持封装字幕, 字体)

* ffmpeg.exe -i ".\视频流入.hevc" -an -c:v copy -i ".\音频流入.aac" -c:a copy -i ".\传统字幕.srt" -c:s copy "封装出.mp4"
* ffmpeg -i ".\视频.hevc" -an -c:v copy -i ".\音轨1.aac" -c:a copy -i ".\音轨2.aac" -c:a copy -i ".\字幕1.ass" -c:s copy -i ".\字幕2.ass" -c:s copy -i ".\字体1.ttf" -c:t copy "封装出.mkv"

### 不同封装格式的字幕格式支持:

[Wikipedia - Subtitle formats support](https://en.wikipedia.org/wiki/Comparison_of_video_container_formats)

### ffmpeg 替换封装中的音频流, itoffset±秒数以对齐:

* ffmpeg.exe -i ".\封装入.mov" -i ".\新音频流入.aac" -c:v copy -map 0:v:0 -map 1:a:0 -c:a copy -itsoffset 0 ".\新封装出.mov"

### ffmpeg: small thread\_queue\_size警告:

* -thread\_queue\_size<(源平均码率kbps+1000)/可调用CPU核心数>

### 批处理: 完成后转换为普通命令窗(不退出):

cmd /k

### +显示Windows版本:

cmd -k

### ffmpeg 从NTSC 3:2 pulldown 60fps源恢复电影标准24fps:

ffmpeg.exe -i ".\60fps\_interlaced\_NTSC\_source.vob" -map 0:v:0 -vf "fieldmatch=order=auto:mode=pc\_n\_ub:field=auto:cthresh=8:combmatch=full:blockx=16:blocky=24:combpel=128,nnedi=weights=C:\[点此下载\nnedi3\_weights.bin](https://github.com/dubhater/vapoursynth-nnedi3/blob/master/src/nnedi3_weights.bin):field=af:nsize=s48x6:nns=n128:qual=slow:etype=mse:pscrn=new3" -fps 24

### x264

### HDR标识

**色域标识**

**色域转换**

--master-display <手动告知播放器拿什么色彩空间解码

DCI-P3: G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(?,1)

bt709: G(15000,30000)B(7500,3000)R(32000,16500)WP(15635,16450)L(?,1)

bt2020: G(8500,39850)B(6550,2300)R(35400,14600)WP(15635,16450)L(?,1)

* 找到HDR元数据中的色彩范围, 确认用以下哪个色彩空间后填上参数
* L的值没有标准, 每个HDR视频元数据里可能都不一样

DCI-P3: G(x0.265, y0.690), B(x0.150, y0.060), R(x0.680, y0.320), WP(x0.3127, y0.329)

bt709: G(x0.30, y0.60), B(x0.150, y0.060), R(x0.640, y0.330), WP(x0.3127,y0.329)

bt2020: G(x0.170, y0.797), B(x0.131, y0.046), R(x0.708, y0.292), WP(x0.3127,y0.329)>

-- cll <和master-display 的L最大值一样>

--colormatrix <照源, 例: gbr bt709 fcc bt470bg smpte170m YCgCo bt2020nc bt2020c smpte2085 ictcp>

--transfer <照源, 例: gbr bt709 fcc bt470bg smpte170m YCgCo bt2020nc bt2020c smpte2085 ictcp>

通用·简单

砍了全部自定义项目, 方便急用但降低了特定画面的压缩率

--rc-lookahead 90 --bframes 12 --b-adapt 2

--me umh --subme 9 --merange 48 --no-fast-pskip --direct auto --weightb

--keyint 360 --min-keyint 5--ref 3

--crf 20 --qpmin 9 --chroma-qp-offset -2

--aq-mode 3 --aq-strength 0.7 --trellis 2

--deblock 0:0 --psy-rd 0.77:0.22 --fgo 10

--nr 8

--fullrange<非7mod x264用, 检查源视频是否使用完整色彩范围>

--me hex --subme 8 --merange 32 --direct auto --weightb

--sliced-threads <降低CPU占用, 减速但时域复杂画面的压缩率可能提高, 参考错误降低>

--vf crop:左,上,右,下/resize:缩放后宽,缩放后高,,,,bicubic/pad:左,上,右,下,直接宽,直接高/hqdn3d:1,1,1,1.5

--zones 0,<片头OP结束帧>,crf=30 --zones<片尾ED开始帧>,<片尾ED结束帧>,crf=30

--seek从第<>帧开始压 --frame 压制<>帧后停止 --fps 元数据没写多少时手动指定帧数

**前瞻进程**

**动态-帧内搜索**

**帧控-参考**

**量化**

**自适量**

**环滤/RDO**

**降噪**

**色彩范围**

**动-帧快速搜索**

**参考冗余优先**

**放/裁/边/降噪滤镜**

**划区压制**

**压制范围**

x264 CLI命令

* x264.exe --rc-lookahead 90 --bframes 12 --b-adapt 2 --me umh --subme 9 --merange 48 --no-fast-pskip --direct auto --weightb --keyint 360 --min-keyint 5 --ref 3 --crf 20 --qpmin 9 --chroma-qp-offset -2 --aq-mode 3 --aq-strength 0.7 --trellis 2 --deblock 0:0 --psy-rd 0.77:0.22 --fgo 10 --nr 8 --output ".\输出.mp4" ".\导入.mp4"

libx264 私有CLI, 兼容libav

* ffmpeg.exe -loglevel 16 -hwaccel auto -y -hide\_banner -i ".\导入.mp4" -c:v libx264 -x264-params "rc-lookahead=90:bframes=12:b-adapt=2:me=umh:subme=9:merange=48:fast-pskip=0:direct=auto:weightb=1:keyint=360:min-keyint=5:ref=3:crf=20:qpmin=9:chroma-qp-offset=-2:aq-mode=3:aq-strength=0.7:trellis=2:deblock=0:0:psy-rd=0.77,0.22:nr=4" -fps\_mode passthrough -c:a copy ".\输出.mp4"

**libx264 私有CLI命令** 上面的 -x264-params 改成 -x264opts, 但没啥意义

**libx264 ffmpeg CLI命令(因版权而重写参数名; 有的ffmpeg发行可能缺参数导致以下命令报错)**

* ffmpeg.exe -loglevel 16 -hwaccel auto -y -hide\_banner -i ".\导入.mp4" -c:v libx264 -bf 12 -b\_strategy 2 -me\_method umh -subq 9 -me\_range 48 -flags2 -fastpskip -directpred 3 -flags2 +wpred -g 360 -keyint\_min 5 -refs 3 -crf 20 -qmin 9 -chromaoffset -2 -aq-mode 3 -aq-strength 0.7 -trellis 2 -deblockalpha 0 -deblockbeta 0 -psy-rd 0.77:0.22 -nr 4 -flags2 +bpyramid -fps\_mode passthrough -c:a copy ".\输出.mp4"

通用·标准

配置起来慢些但自定义范围广。由于x264参数不多, 所以这套参数足以涵盖高画质高压-录像-动画情形

--rc-lookahead <3×帧率> --bframes 12 --b-adapt 2

--me umh --subme <电影10~11, 动漫9 (11仅7mod)> --merange <快20, 高压48, 4的倍数> --no-fast-pskip --direct auto --weightb

--keyint <8~10×帧率> --min-keyint <1增加IDR, 5常用> --ref 3

--crf 19 --qpmin 9 --chroma-qp-offset <常用-1~-2, 动漫-3~-5>

--aq-mode 3 --aq-strength <一般0.7, 原画1.1> --trellis 2

--deblock <一般0:0, 原画-1:-1> --psy-rd<动漫0.4~.6:0.1~.15, 录像0.7~1.3:0.12~.2> --fgo 12

--nal-hrd --vbv-bufsize <最大kbps每秒> --vbv-maxrate <bufsize倍数的kbps>

--fullrange<非7mod x264用, 检查源视频是否使用完整色彩范围>

--sliced-threads <降CPU占用, 减速但时域复杂画面的压缩率可能提高, 参考错误降低>

--vf crop:左,上,右,下/resize:缩放后宽,缩放后高,,,,bicubic/pad:左,上,右,下,直接宽,直接高/hqdn3d:1.1,1.1,1.1,1.1

--zones 0,<片头OP结束帧>,crf=30 --zones<片尾ED开始帧>,<片尾ED结束帧>,crf=30

--seek从第<>帧开始压 --frame 压制<>帧后停止 --fps 元数据没写多少时手动指定帧数

**前瞻进程**

**动态-帧内搜索**

**帧控-参考**

**量化**

**自适应量化**

**环滤/RDO**

**CRF-VBR压缩**

**色彩范围**

**参考冗余优先**

**放/裁/边/降噪**

**参数划区压制**

**压制范围**

x264 CLI命令

* x264.exe --me umh --bframes 12 --b-adapt 2 --subme ? --merange ? --no-fast-pskip --direct auto --weightb --keyint ? --min-keyint ? --ref 3 --rc-lookahead ? --crf 19 --qpmin 9 --chroma-qp-offset ? --aq-mode 3 --aq-strength ? --trellis 2 --deblock ? --psy-rd ? --fgo 12 --output ".\输出.mp4" ".\导入.mp4"

libx264 私有CLI, 兼容libav, 不支持fgo

* ffmpeg.exe -loglevel 16 -hwaccel auto -y -hide\_banner -i ".\导入.mp4" -c:v libx264 -x264-params " bframes=12:b-adapt=2:me=umh:subme=〇:merange=〇:fast-pskip=0:direct=auto:weightb=1:keyint=〇:min-keyint=〇:ref=3:rc-lookahead=〇:crf=19:qpmin=9:chroma-qp-offset=〇:aq-mode=3:aq-strength=〇:trellis=2:deblock=0:-1:psy-rd=〇,〇:nr=4" -fps\_mode passthrough -c:a copy ".\输出.mp4"

**libx264 私有CLI, 不支持fgo** 上面的 -x264-params 改成 -x264opts, 但没啥意义

**libx264 ffmpeg CLI命令(因版权而重写参数名; 有的ffmpeg发行可能缺参数导致以下命令报错)**

* ffmpeg.exe -loglevel 16 -hwaccel auto -y -hide\_banner -i ".\导入.mp4" -c:v libx264 -me\_method umh -subq 〇 -me\_range 〇 -flags2 -fastpskip -directpred 3 -flags2 +wpred -g 〇 -keyint\_min 〇 -bf 12 -b\_strategy 2 -refs 3 -crf 19 -qmin 9 -chromaoffset 〇 -aq-mode 3 -aq-strength 〇 -trellis 2 -deblockalpha 0 -deblockbeta -1 -psy-rd 〇:〇 -nr 4 -flags2 +bpyramid -fps\_mode passthrough -c:a copy ".\输出.mp4"

剪辑素材存档

加强无损压缩, 降低有损压缩, 增加IDR帧数量. 建议YUV4:2:2或4:4:4 8~10bit

--bframes 12 --b-adapt 2

--me esa --subme <电影10~11, 动漫9 (11仅7mod)> --merange <快速40, 高压48> --no-fast-pskip --direct auto --weightb

--keyint <5~8×帧率> --min-keyint 1 --ref 3 --sliced-threads

--crf 17 --tune grain

--trellis 2 --fgo 15

--zones 0,<片头/OP结束帧>,crf=30 --zones<片尾/ED开始帧>,<片尾/ED结束帧>,crf=30

--seek从第<>帧开始压 --frame 压制<>帧后停止 --fps 元数据没写多少时手动指定帧数

**前瞻进程**

**动态-帧内搜索**

**帧控-参考**

**量化-主控**

**自适-RDO**

**划区压制**

**压制范围**

x264 CLI命令

* x264.exe --bframes 12 --b-adapt 2 --me esa --subme 〇 --merange 〇 --no-fast-pskip --direct auto --weightb --keyint 〇 --min-keyint 1 --ref 3 --crf 17 --tune grain --trellis 2 --fgo 15 --output ".\输出.mp4" ".\导入.mp4"

libx264 私有CLI, 兼容libav, 不支持fgo

* ffmpeg.exe -loglevel 16 -hwaccel auto -y -hide\_banner -i ".\导入.mp4" -c:v libx264 -x264-params "me=esa:subme=〇:merange=〇:fast-pskip=0:direct=auto:weightb=1:keyint=〇:min-keyint=1:bframes=12:b-adapt=2:ref=3:crf=17:trellis=2" -fps\_mode passthrough -c:a copy ".\输出.mp4"

**libx264 私有CLI, 不支持fgo** 上面的 -x264-params 改成 -x264opts, 但没啥意义

**libx264 ffmpeg CLI命令(因版权而重写参数名; 有的ffmpeg发行可能缺参数导致以下命令报错)**

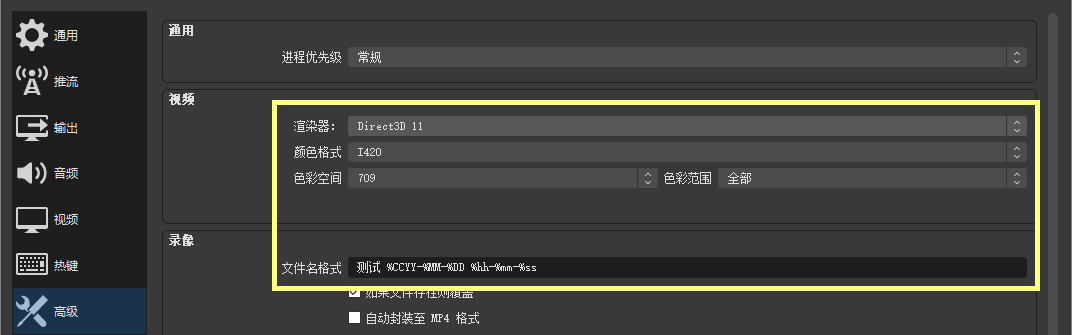
* ffmpeg.exe -loglevel 16 -hwaccel auto -y -hide\_banner -i ".\导入.mp4" -c:v libx264 -bf 12 -b\_strategy 2 -me\_method esa -subq 〇 -me\_range 〇 -flags2 -fastpskip -directpred 3 -flags2 +wpred -g 〇 -keyint\_min 1 -refs 3 -crf 17 -tune grain -trellis 2 -flags2 +bpyramid -fps\_mode passthrough -c:a copy ".\输出.mp4"

# OBS录屏

打开OBS🡪文件🡪设置, 一般录屏的预设置如下：

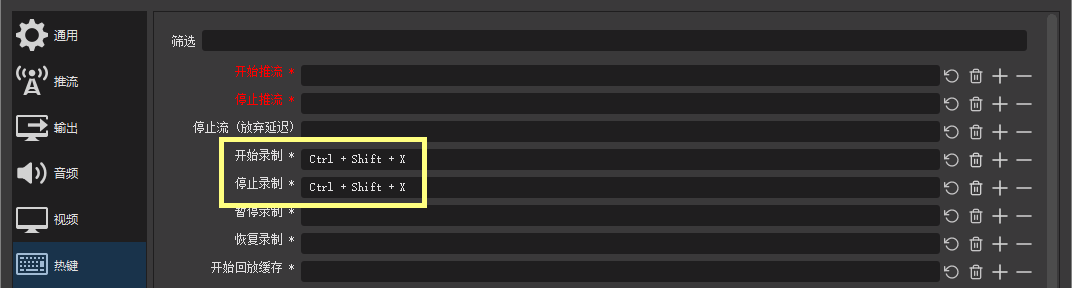
**高级**:

颜色格式-色彩空间-色彩范围-文件名格式, 一般除文件名外和图中保持一致就行



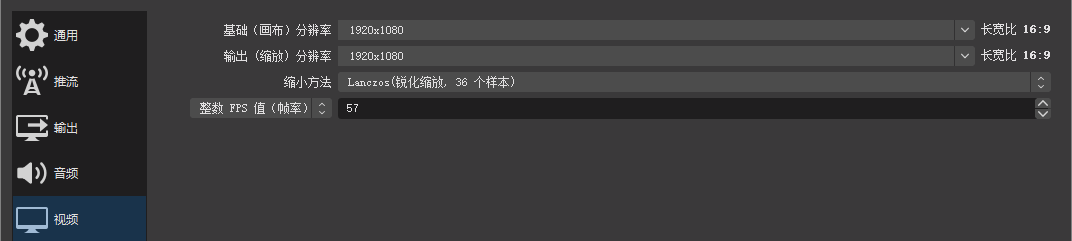
**热键**:

开始/停止录制, 看个人习惯, 如Ctrl+Shift+X

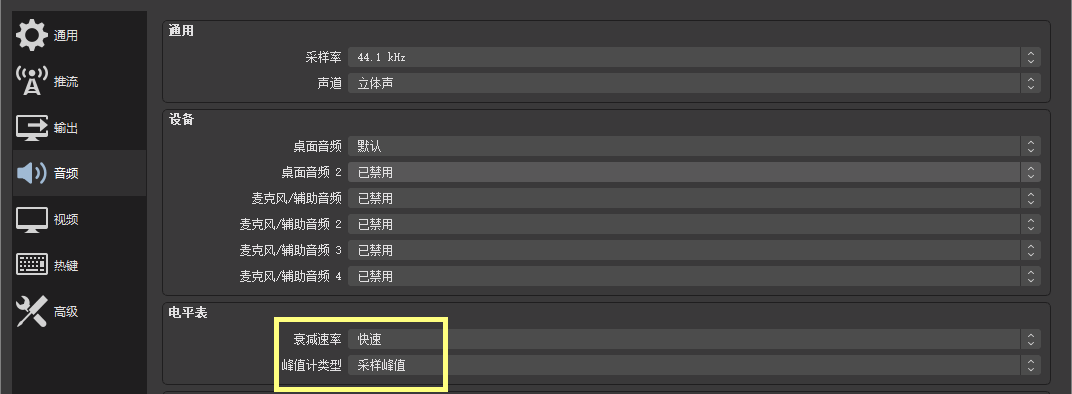


**帧率**:一般设在57到58, 损失一点帧数换取更低CPU占用和一点画质

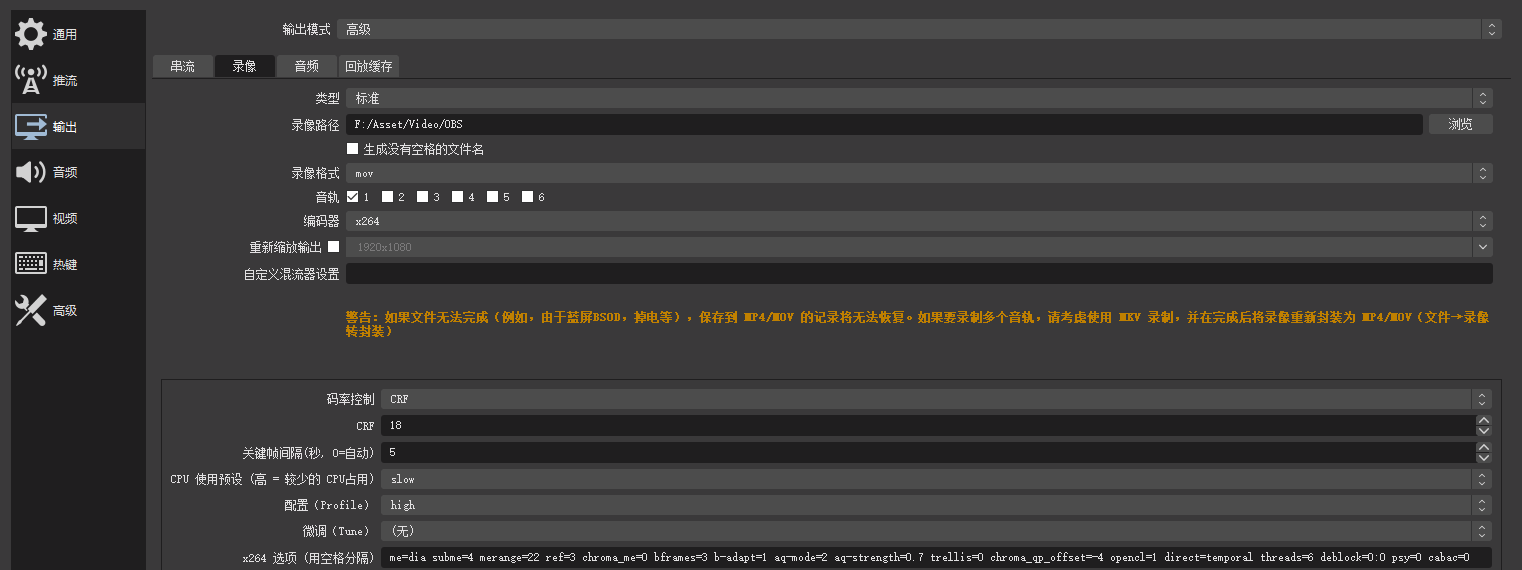
**分辨率**:一般设在1920x1040, 去掉一点宽, 道理同上



**音频**:奇怪的是占用CPU更低的电平和采样峰值反而要更清晰...



**参数**:设定输出路径, 然后将下面需要的参数粘贴到这里



录屏10fps高压 - 音乐/助眠/棋牌/B-roll

码率控制： CRF 关键帧间隔： 3（这里是秒）

CPU 使用预设： slow CRF： 20

Profile： high tune： stillimage

设定-高级-色彩空间-色度采样： i420

参数： me=hex subme=7 chroma\_me=0 bframes=12 b-adapt=1 ref=2 aq-mode=0 aq-strength=0 deblock=0:-1 trellis=2 chroma\_qp\_offset=-4 cabac=1 psy=0 opencl=1 fgo=15

* 由于画面变化极小也没有突发情况, 所以用不到VBR模式（VBV参数）

录屏28fps高压 - 聊天/演奏/小游戏/装机/发布会

码率控制： VBR 关键帧间隔： 8（这里是秒）

比特率： 9000kbps CPU 使用预设： medium

缓冲大小： 默认 Profile： main

CRF: 19 tune： film

设定-高级-色彩空间-色度采样： i420

参数： me=umh subme=6 chroma\_me=1 bframes=5 b-adapt=1 ref=2 aq-mode=1 aq-strength=0.7 deblock=0:0 trellis=2 direct=temporal opencl=1 fgo=15

* 画面有动态, 且以上情况里的背景一定是静态的, 所以动搜增强了很多
* 画面可能有突发情况, 所以用了VBR模式
* 增加了IDR帧时间来给码率让位

录屏55fps低压 - FPS-STG-赛车-3A游戏/演唱会

码率控制： VBR/ABR 关键帧间隔： 5（这里是秒）

比特率： 35000kbps CPU 使用预设： veryfast

缓冲大小： 3500 Profile： 无

CRF： 18 tune： film

参数： me=hex subme=4 me\_range=12 chroma\_me=0 bframes=3 b-adapt=1 ref=3 aq-mode=2 aq-strength=0.9 deblock=0:0 trellis=0 deadzone-inter=8 deadzone-intra=5 direct=temporal cabac=0 opencl=1 nr=10 fgo=15

* deadzone-inter=8 deadzone-intra=5这两个参数必须在trellis<2时使用, 否则画面会变得很脏
* 为进一步限制码率, 使用了降噪nr功能
* 本方案没有考虑游戏的突发高占用情况, 可能会在大量AI, 大量效果等情况下造成卡顿；虽然这是游戏本身就把CPU占满的情况

录屏100FPS - x264低压

码率控制： CRF/ABR 关键帧间隔： 6（秒）

CPU 使用预设： fast CRF/比特率： 18/9000 kbps（或直播平台/网速上限）

Profile： 无 tune： 无

参数： me=hex me\_range=12 subme=3 chroma\_me=0 bframes=3 b-adapt=1 ref=3 aq-mode=0 psy=0 mbtree=0 cabac=0 mixed\_refs=0 deadzone-inter=8 deadzone-intra=5 deblock=0:0 trellis=0 direct=temporal ref=3 opencl=1 fgo=15

* 如果以上参数录制的视频仍然卡顿, CPU占用高的话, 就需要限制渲染帧率, 开fps\_mode, 降低画质设定, 升级/超频CPU/内存, 用另一台电脑压制, 用采集卡, 用图灵RTX NVENC硬件编码, 用帕斯卡GTX Nvenc硬件编码, 用x264 ABR模式等其它录制方案
* 以上方案列表按照录屏画质, 从高到低排序
* Potplayer播放器的视频色彩解析有问题, 建议用mpv预览

# Twitch 3A直播压制机 VBR 1080p60<8000kbps

## **4:4:4 D3D11 Limited; 超频R7 1800x+3000Mhz CL15 2x4G** *已验证***：**

码率控制： VBR 关键帧间隔： 2（秒, Twitch标准）

比特率： 7800kbps CPU 使用预设： veryfast

缓冲大小： 3500 Profile： 无

CRF： 22 tune： 无

参数： me=umh subme=5 me\_range=16 chroma\_me=0 bframes=5 b-adapt=1 ref=3 deblock=0:0 trellis=2 aq-mode=1 aq-strength=1.1 cabac=1

## **4:4:4 D3D11 Limited; 4.0Ghz R7 2700x+3000Mhz CL15 2x4G** *未验证***：**

码率控制： VBR 关键帧间隔： 2（秒, Twitch标准）

比特率： 7800kbps CPU 使用预设： fast

缓冲大小： 3500 Profile： 无

CRF： 22 tune： 无

参数： me=umh subme=7 me\_range=16 chroma\_me=0 bframes=5 b-adapt=1 ref=3 deblock=0:0 trellis=2 aq-mode=2 aq-strength=1.1 direct=auto cabac=1 nr=15 chroma\_qp\_offset=-2

## **4:4:4 D3D11 Limited;默频R7 3800x+3000Mhz CL15 2x8G CR-2T** *未验证***：**

码率控制： VBR 关键帧间隔： 2（秒, Twitch标准）

比特率： 7830kbps CPU 使用预设： medium

缓冲大小： 3500 Profile： 无

CRF： 22 tune： 无

参数： me=umh subme=9 me\_range=16 chroma\_me=0 bframes=6 b-adapt=2 ref=3 deblock=0:0 trellis=2 aq-mode=2 aq-strength=1.1 direct=auto cabac=1 nr=15 chroma\_qp\_offset=-4 b\_deterministic=0

## **4:4:4 D3D11 Limited; 4.2Ghz R7 5800x+3000Mhz CL15 2x16G CR-1T** *未验证***：**

码率控制： VBR 关键帧间隔： 2（秒, Twitch标准）

比特率： 7830kbps CPU 使用预设： medium

缓冲大小： 3500 Profile： 无

CRF： 22 tune： 无

参数： me=umh subme=9 me\_range=20 chroma\_me=0 bframes=7 b-adapt=2 ref=3 deblock=0:-1 trellis=2 aq-mode=3 aq-strength=1.1 direct=auto cabac=1 nr=15 chroma\_qp\_offset=-4 b\_deterministic=0