# pip安装报平台不支持

# [pip安装报错：is not a supported wheel on this platform](http://www.cnblogs.com/nice-forever/p/5371906.html)

可能的原因1：安装的不是对应python版本的库，下载的库名中cp27代表python2.7,其它同理。

可能的原因2：这个是我遇到的情况（下载的是对应版本的库，然后仍然提示不支持当前平台）

　　我下载到的numpy库文件名：

* numpy-1.10.4+mkl-cp27-cp27m-win32.whl

　　使用pip安装(在命令行中):

* pip install numpy-1.10.4+mkl-cp27-cp27m-win32.whl

　　报错：\*\*\*  is not a supported wheel on this platform，通过在stackoverflow上的一个帖子成功解决问题。

方法：在shell中输入

import pip; print(pip.pep425tags.get\_supported())

可以获取到pip支持的文件名还有版本。现在pip改版，如果报错可以改成这样输入：

from pip.\_internal.pep425tags import get\_supported;

get\_supported()

我这里如下：

[复制代码](javascript:void(0);)

>>import pip; print(pip.pep425tags.get\_supported())  
[('cp27', 'none', 'win32'), ('py2', 'none', 'win32'), ('cp27', 'none', 'any'), ('cp2', 'none', 'any'), ('cp26', 'none', 'any'), ('cp25', 'none', 'any'), ('cp24', 'none', 'any'), ('cp23', 'none', 'any'), ('cp22', 'none', 'any'), ('cp21', 'none', 'any'), ('cp20', 'none', 'any'), ('py27', 'none', 'any'), ('py2', 'none', 'any'), ('py26', 'none', 'any'), ('py25', 'none', 'any'), ('py24', 'none', 'any'), ('py23', 'none', 'any'), ('py22', 'none', 'any'), ('py21', 'none', 'any'), ('py20', 'none', 'any')]

[复制代码](javascript:void(0);)

　　通过这里可以发现上面下载的文件名格式是不支持的，修改为：numpy-1.10.4+mkl-cp27-none-win32.whl即可成功安装。

　　其它的库也同理可以成功安装，不过也请注意库的依赖。

　　（参考帖子网址：http://stackoverflow.com/questions/28107123/cannot-install-numpy-from-wheel-format?rq=1）

　　（python库下载地址(库很丰富，下载速度也很快)：http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy）

# 2、安装tensorflow时报GLIBC\_2.14找不到

# Linux/Centos下/lib64/libc.so.6: version `GLIBC\_2.14' not found问题

前天，在Centos的某个版本下编译了一个可执行程序，复制到另外一个Centos环境下去执行，结果出现了以下错误：

/lib64/libc.so.6: version `GLIBC\_2.14' not found

貌似是一个很普遍的错误，去网上搜集了相关的资料并整理了一下

出现这种错误表明程序运行需要GLIBC\_2.14，但是系统中却并不存在，因此可以先用strings命令查看下系统中的GLIBC版本

strings /lib64/libc.so.6 | grep GLIBC

发现系统中最高只支持GLIBC\_2.12，解决这个问题有多种方法。

在你准备升级GLIBC库之前，你要好好思考一下，  
你真的要升级GLIBC么？  
你知道你自己在做什么么？  
http://baike.baidu.com/view/1323132.htm?fr=aladdin  
  
glibc是gnu发布的libc库，即c运行库。glibc是linux系统中最底层的api，几乎其它任何运行库都会依赖于glibc。glibc除了封装linux操作系统所提供的系统服务外，它本身也提供了许多其它一些必要功能服务的实现…  
总的来说，不说运行在linux上的一些应用，或者你之前部署过的产品，就是很多linux的基本命令，比如cp, rm, ll之类，都得依赖于它  
网上很多人有惨痛教训，甚至升级失败后系统退出后无法重新进入了。。。。。。  
  
对于CentOS这样的系统，为了追求稳定性（这个值得商榷）往往各种库版本都很低，比如6.5甚至7.0自带的还是glibc2.12, 而ubuntu 14.04带glibc2.19  
如果升级基本C运行库到一个太新的版本，可能会影响CentOS的运行。所以大家如果遇到CentOS基本库的问题，影响了自己程序的运行，应该可以考虑：  
1. 在低版本的系统编译自己的产品，如果自己的产品确实不需要新版才支持的新特性  
2. 用版本高的系统来编译，比如ubuntu，和centos的新版，但可能需要部署到较低版本，那么可以考虑用mock等技术制作更好的安装包，把依赖打入包内  
3.利用容器技术，如Docker，在低版本的操作系统内，轻量级的隔离出一个虚拟运行环境，适应你的程序。  
好在我遇到的问题是glibc2.15就满足要求升级后暂时没发现问题，所以大家可以参考我的方法：  
首先查看现有的情况，在CentOS6.5下

ll /lib64/libc.so.6

libc.so.6是一个软连接，当前的glibc是2.12版本,我遇到的是GLIBC\_2.15找不到的问题，所以需至少升级到2.15  
首先，从网上下载glibc 2.15的rpm安装包，但这个不容易，因为.rpm针对的是centOS和redhat，高版本安装包很少见。也可以直接从其他系统上拷一个编译好的文件libc.so.6（对应glibc 2.15或者更高的），不过最保险的方式就是下载源代码在本地编译一次（有的人实在编译不成功，那也只能从别的地方找一份了）  
各个版本的glibc可以从http://ftp.gnu.org/gnu/glibc/找，包括其插件glibc-port  
最新到2.20，我保守的选择2.15  
对于低版本glibc，还有glibc-linuxthreads-2.x需要编译，可参考很多网上文档，但2.15没有，所以不用了

**[plain]** [view plain](http://blog.csdn.net/sole_cc/article/details/51415940" \o "view plain) [copy](http://blog.csdn.net/sole_cc/article/details/51415940" \o "copy)

1. wget http://ftp.gnu.org/gnu/glibc/glibc-2.15.tar.gz
3. wget http://ftp.gnu.org/gnu/glibc/glibc-ports-2.15.tar.gz
5. tar -xvf  glibc-2.15.tar.gz
7. tar -xvf  glibc-ports-2.15.tar.gz
9. mv glibc-ports-2.15 glibc-2.15/ports
11. mkdir glibc-build-2.15

14. cd glibc-build-2.15

17. ../glibc-2.15/configure  --prefix=/usr --disable-profile --enable-add-ons --with-headers=/usr/include --with-binutils=/usr/bin

20. make
21. make install

如果提示install成功，去看glibc所在的共享库：

可以看到2.12的旧库文件还在，多了2.15版本的库文件，而且软链接文件全部指向了2.15版本。

**[plain]** [view plain](http://blog.csdn.net/sole_cc/article/details/51415940" \o "view plain) [copy](http://blog.csdn.net/sole_cc/article/details/51415940" \o "copy)

1. -rwxr-xr-x  1 root root 1921096 Aug 30 02:16 /lib64/libc-2.12.so
2. -rwxr-xr-x  1 root root 9801632 Sep 25 13:46 /lib64/libc-2.15.so
3. lrwxrwxrwx. 1 root root      18 May 19 18:51 /lib64/libcap-ng.so.0 -> libcap-ng.so.0.0.0
4. -rwxr-xr-x. 1 root root   18672 Jun 25  2011 /lib64/libcap-ng.so.0.0.0
5. lrwxrwxrwx. 1 root root      14 May 19 18:51 /lib64/libcap.so.2 -> libcap.so.2.16
6. -rwxr-xr-x  1 root root   19016 Dec  8  2011 /lib64/libcap.so.2.16
7. lrwxrwxrwx. 1 root root      19 May 19 18:57 /lib64/libcgroup.so.1 -> libcgroup.so.1.0.40
8. -rwxr-xr-x  1 root root   97016 Dec  9  2013 /lib64/libcgroup.so.1.0.40
9. -rwxr-xr-x  1 root root  197064 Aug 30 02:16 /lib64/libcidn-2.12.so
10. -rwxr-xr-x  1 root root  267972 Sep 25 13:46 /lib64/libcidn-2.15.so
11. lrwxrwxrwx  1 root root      15 Sep 25 13:52 /lib64/libcidn.so.1 -> libcidn-2.15.so
12. lrwxrwxrwx. 1 root root      17 May 19 18:51 /lib64/libcom\_err.so.2 -> libcom\_err.so.2.1
13. -rwxr-xr-x  1 root root   17256 Nov 22  2013 /lib64/libcom\_err.so.2.1
14. -rwxr-xr-x  1 root root   40400 Aug 30 02:16 /lib64/libcrypt-2.12.so
15. -rwxr-xr-x  1 root root  142947 Sep 25 13:46 /lib64/libcrypt-2.15.so
16. lrwxrwxrwx. 1 root root      22 May 19 18:57 /lib64/libcryptsetup.so.1 -> libcryptsetup.so.1.1.0
17. -rwxr-xr-x  1 root root   97072 Jun 22  2012 /lib64/libcryptsetup.so.1.1.0
18. lrwxrwxrwx  1 root root      16 Sep 25 13:52 /lib64/libcrypt.so.1 -> libcrypt-2.15.so
19. lrwxrwxrwx  1 root root      12 Sep 25 13:52 /lib64/libc.so.6 -> libc-2.15.so

有些人会在make install后出现error。这儿error我没去细究，此时可以查看一下系统此时的GLIBC版本，参考一开始的做法。如果版本未升级，我们只能手动安装一下：

首先make是成功了，那么我们会发现build目录下编译出了一个新的libc.so.6 （/glibc-build-2.15/libc.so.6， 我们会发现这实际上也是一个软连接，真实的lib文件是libc.so， 输出

**[plain]** [view plain](http://blog.csdn.net/sole_cc/article/details/51415940" \o "view plain) [copy](http://blog.csdn.net/sole_cc/article/details/51415940" \o "copy)

1. $ ll  libc.so.6
2. lrwxrwxrwx 1 root root 7 Sep 23 07:41 libc.so.6 -> libc.so
3. [usr@linux glibc-build-2.15]$ strings libc.so | grep GLIBC
4. GLIBC\_2.2.5
5. GLIBC\_2.2.6
6. GLIBC\_2.3
7. GLIBC\_2.3.2
8. GLIBC\_2.3.3
9. GLIBC\_2.3.4
10. GLIBC\_2.4
11. GLIBC\_2.5
12. GLIBC\_2.6
13. GLIBC\_2.7
14. GLIBC\_2.8
15. GLIBC\_2.9
16. GLIBC\_2.10
17. GLIBC\_2.11
18. GLIBC\_2.12
19. GLIBC\_2.13
20. GLIBC\_2.14
21. GLIBC\_2.15
22. GLIBC\_PRIVATE

这是我们需要的lib了，然后去更新系统的库。  
这里要注意，更新系统里的链接（我的是/lib64/libc.so.6) 很容易出错，我不清楚有没有更好的办法，一般都是删除旧链接，建立新链接  
但删除旧链接后，很多命令直接不能用了，因为此时系统中不到glibc的库了。这个时候就需要临时指定一个glibc库，方法如下（libc.so改个名以便好与以后更新的其他版本区分）：

**[plain]** [view plain](http://blog.csdn.net/sole_cc/article/details/51415940" \o "view plain) [copy](http://blog.csdn.net/sole_cc/article/details/51415940" \o "copy)

1. cp /\*\*\*\*/glibc-build-2.15/libc.so /lib64/libc-2.15.so
2. rm -rf /lib64/libc.so.6
3. LD\_PRELOAD=/lib64/libc-2.15.so ln -s/lib64/libc-2.15.so  lib64/libc.so.6

更新连接完毕，然后

**[plain]** [view plain](http://blog.csdn.net/sole_cc/article/details/51415940" \o "view plain) [copy](http://blog.csdn.net/sole_cc/article/details/51415940" \o "copy)

1. $ strings /lib64/libc.so.6 | grep GLIBC
2. GLIBC\_2.2.5
3. GLIBC\_2.2.6
4. GLIBC\_2.3
5. GLIBC\_2.3.2
6. GLIBC\_2.3.3
7. GLIBC\_2.3.4
8. GLIBC\_2.4
9. GLIBC\_2.5
10. GLIBC\_2.6
11. GLIBC\_2.7
12. GLIBC\_2.8
13. GLIBC\_2.9
14. GLIBC\_2.10
15. GLIBC\_2.11
16. GLIBC\_2.12
17. GLIBC\_2.13
18. GLIBC\_2.14
19. GLIBC\_2.15
20. GLIBC\_PRIVATE

说明连接更新成功，再编译的话，GLIBC\_2.15及以下版本的依赖问题就不会出现了。

http://love.junzimu.com/archives/2269

# 其它问题参见：

# 深度学习框架\_TensorFlow安装.docx