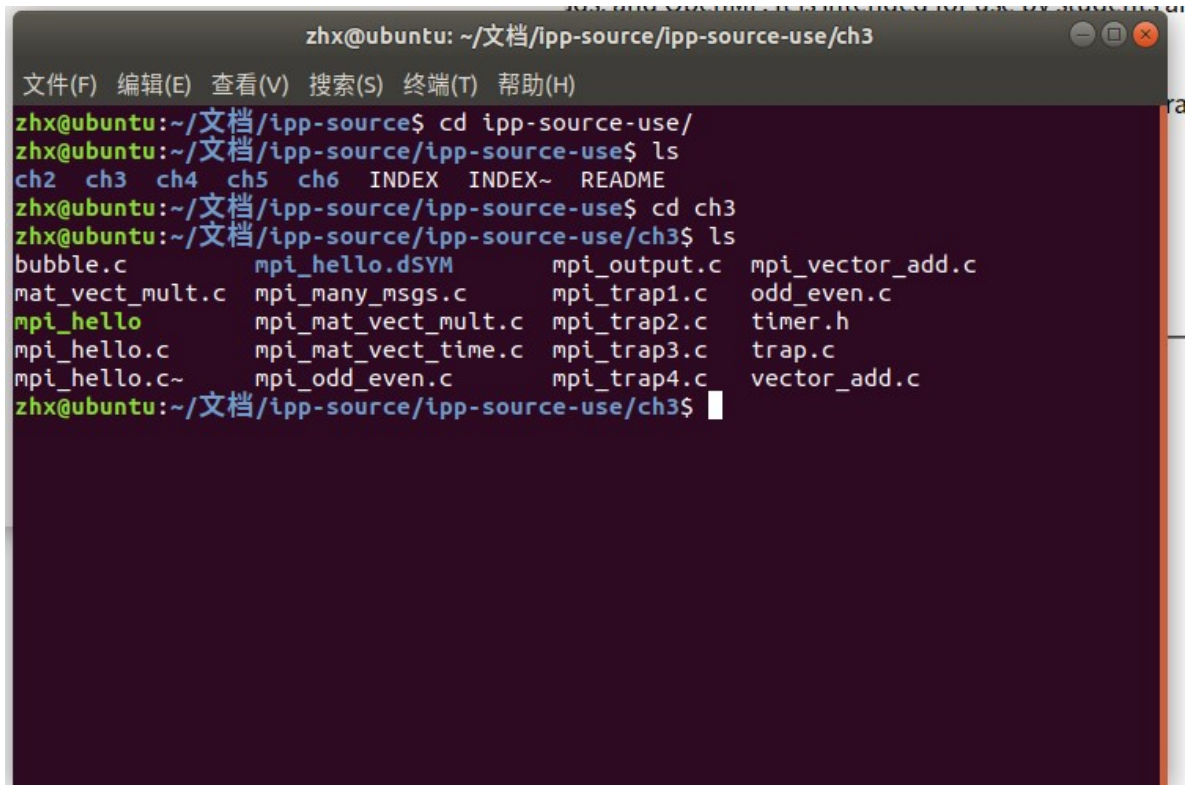


Homework 1

2018011365 张鹤潇

Q1.

截图如下，系统版本ubuntu-18.04.3.



```
zhx@ubuntu: ~/文档/ipp-source/ipp-source-use/ch3
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
zhx@ubuntu:~/文档/ipp-source$ cd ipp-source-use/
zhx@ubuntu:~/文档/ipp-source/ipp-source-use$ ls
ch2  ch3  ch4  ch5  ch6  INDEX  INDEX~  README
zhx@ubuntu:~/文档/ipp-source/ipp-source-use$ cd ch3
zhx@ubuntu:~/文档/ipp-source/ipp-source-use/ch3$ ls
bubble.c      mpi_hello.dSYM      mpi_output.c      mpi_vector_add.c
mat_vect_mult.c  mpi_many_msgs.c      mpi_trap1.c        odd_even.c
mpi_hello      mpi_mat_vect_mult.c  mpi_trap2.c        timer.h
mpi_hello.c     mpi_mat_vect_time.c  mpi_trap3.c        trap.c
mpi_hello.c~    mpi_odd_even.c       mpi_trap4.c        vector_add.c
zhx@ubuntu:~/文档/ipp-source/ipp-source-use/ch3$
```

Q2.

根据核编号与 $n\%p$ 的大小关系分配，以使得任务分配尽可能均匀。代码如下(in python):

```
def get_range(n, p, my_rank):
    quotient, remainder = n//p, n%p
    if my_rank < remainder:
        cnt = quotient+1
        my_first_i = my_rank*cnt
    else:
        cnt = quotient
        my_first_i = my_rank*cnt+remainder
    my_last_i = my_first_i+cnt
    return my_first_i, my_last_i
```

Q3.

a.0号核接收和加法操作 $P - 1$ 次.

b.0号核接受和加法操作 $\lceil \log_2 P \rceil$ 次.

P	a	b
2	1	1
4	3	2
8	7	3
16	15	4
32	31	5
64	63	6
128	127	7
256	255	8
512	511	9
1024	1023	10

Q4.

两种并行方式都有。在算法的该部分，我们将 $\lceil \log_2 P \rceil + 1$ 种任务(对应于参与 $0, 1, \dots, \lceil \log_2 P \rceil$ 次接受&求和)，分配给 P 个核心去执行。存在执行相同任务而数据不同的核心，即所谓数据并行；也存在执行不同任务的核心，即所谓任务并行。

Q5.

我曾在小学期用多进程(线程)优化python网络爬虫，将网页内容爬取的速度提高了10倍以上。

代码首先获取中央处理器内核个数，然后启动相应的进程个数，在每进程启动多个线程爬虫，每个线程负责读取网址列表中的一部分网页。

考虑到，如果将爬虫队列存储在内存中，则多进程数据写入难以处理；为了解决这一问题，我把爬虫队列转移到MongoDB当中。用MongoDB的内建队列保存url，并调用其接口自动管理数据写入。当所有线程的任务执行完毕后，网页爬取工作就完成了。

由于下载带宽是有限的，添加过多新线程无法加快下载速度；要想获得更好性能的爬虫，可能需要在多台服务器上分布式部署爬虫。

这是典型的数据并行，因为它在多组网页上并行执行相同的爬取算法。