Imports:

```
from sys import argv
import os, multiprocessing, signal
import time, random, threading
#!/bin/bash
```

/
**********/

Files:

```
f = open(path, 'w')
f.close(), f.write(str)
os.chmod(path, int(per, 8))
os.chmod(path, 0o755)

f.write(bytearray({list name}))
int.from_bytes(f.read(1), byteorder='big')
f.seek(j, [0, 1, 2])
```

/
**********/

Processes:

```
os.getpid(), os.getppid()
sys.exit(0), os.fork()
os.waitpid(pid, 0) // 0 if process is running
os.execl('./a.out', './a.out', arg1)
os.kill(pid, signal.SIGKILL)
```

/
**********/

Socket:

```
s = socket.socket()
host = socket.gethostname()
port = 8889

s.connect((host, port))
s.send(msg.encode())
ans = s.recv(1024).decode()

class client():
    def __init__(self, socket, address):
        self.sock = socket
        self.addr = address
        self.run()

    def run(self):
        while(1):
        msg = c.recv(1024).decode()
        self.sock.send((str(a +
        b)).encode())
```

```
s = socket.socket()
host = socket.gethostname()
port = 8889
s.bind((host, port))

s.listen(5)
while clients < MAX_CLIENT:
    c, addr = s.accept()
    print('Got connection from', addr)
    client(c, addr)
```

/
**********/

Pipe:

```
r, w = os.pipe()
p = os.fdopen(w, 'w')
os.close(r) // w

os.write(w, bytes(str(rand), 'ascii'))
os.read(r, 1).decode('ascii')

os.mkfifo('clientToServer')
os.mkfifo('serverToClient')
```

/
**********/

Signal:

```
signal.signal(signal.SIGINT, signal_handler)
def signal_handler(signal, frame):
```

/
**********/

Thread:

```
th = threading.Thread(target=f, args=(ind,))
th.start()
```

/
**********/

Other:

```
random.seed(integer[pid])
random.randint(a, b)
time.sleep(s) - time.time()
 os.system(command)
lock = threading.Lock()
mkdir -p
useradd -m
```