脚本阅读报告

第一组：张扬、张耀、张光云、梁健恒、黄旸珉

目录
Contents

1

辅助函数

除生成证书、生成脚本、生成配置文件和主函数的函数，主要包含在前220行的代码中。主要功能为用户呈现帮助菜单、输出登陆信息、登陆警告以及进行基本的逻辑检查

帮助菜单界面

help()

echo $1输出传递给函数的第一个参数。然后将help函数中不同指令对应的不同操作输出到屏幕上，方便用户选择。

print_result()

函数是在build-chain.sh成功执行后输出成功搭建的4节点联盟链的信息。函数调用了LOG-INFO分别输出了FISCO-BCOS路径、起始端口、服务器ip、状态类型、RPC监听ip、输出目录、CA Key存储目录等信息。

! -z ${docker_mode}

! -z $ip_file

! -z ${pkcs12_passwd}

! -z $guomi_mode

这四段代码作用是当字段不为空时，分别输出对应的信息。最后调用LOG-INFO输出“All completed”表示信息输出结束，并输出文件存储的位置。

parse_params()

这个函数就是对用户在help界面的输入进行语法分析。getopts函数根据用户输入的不同选择输出不同的结果。比如用户输入i则将系统的监听ip输出到屏幕上。在用户输入p选项时，

${#port_start[@]} -ne 3

代码会判断当前端口是否等于3，如果不等于则会调用LOG-WARN函数输出起始端口错误的信息。另外当用户输入C选项时，

-z$(grep'^[[:digit:]]*$'<<<"${chain_id}")

会检查用户输入的链id是否为正整数，如果id不是正整数则会调用LOG-WARN函数输出错误信息并提示用户id不是正整数。 。

```
Usage:
    -l <IP list>
    -f <IP list file>
    -e <FISCO-BCOS binary path>
    -o <Output Dir>
    -p <Start Port>
    -i <Host ip>
    -v <FISCO-BCOS binary version>
    -d <docker mode>
    -s <State type>
    -S <Storage type>
    -c <Consensus Algorithm>
    -C <Chain id>
    -g <Generate guomi nodes>
    -z <Generate tar packet>
    -t <Cert config file>
    -T <Enable debug log>
    -F <Disable log auto flush>
    -h Help
```

输出警告、错误信息

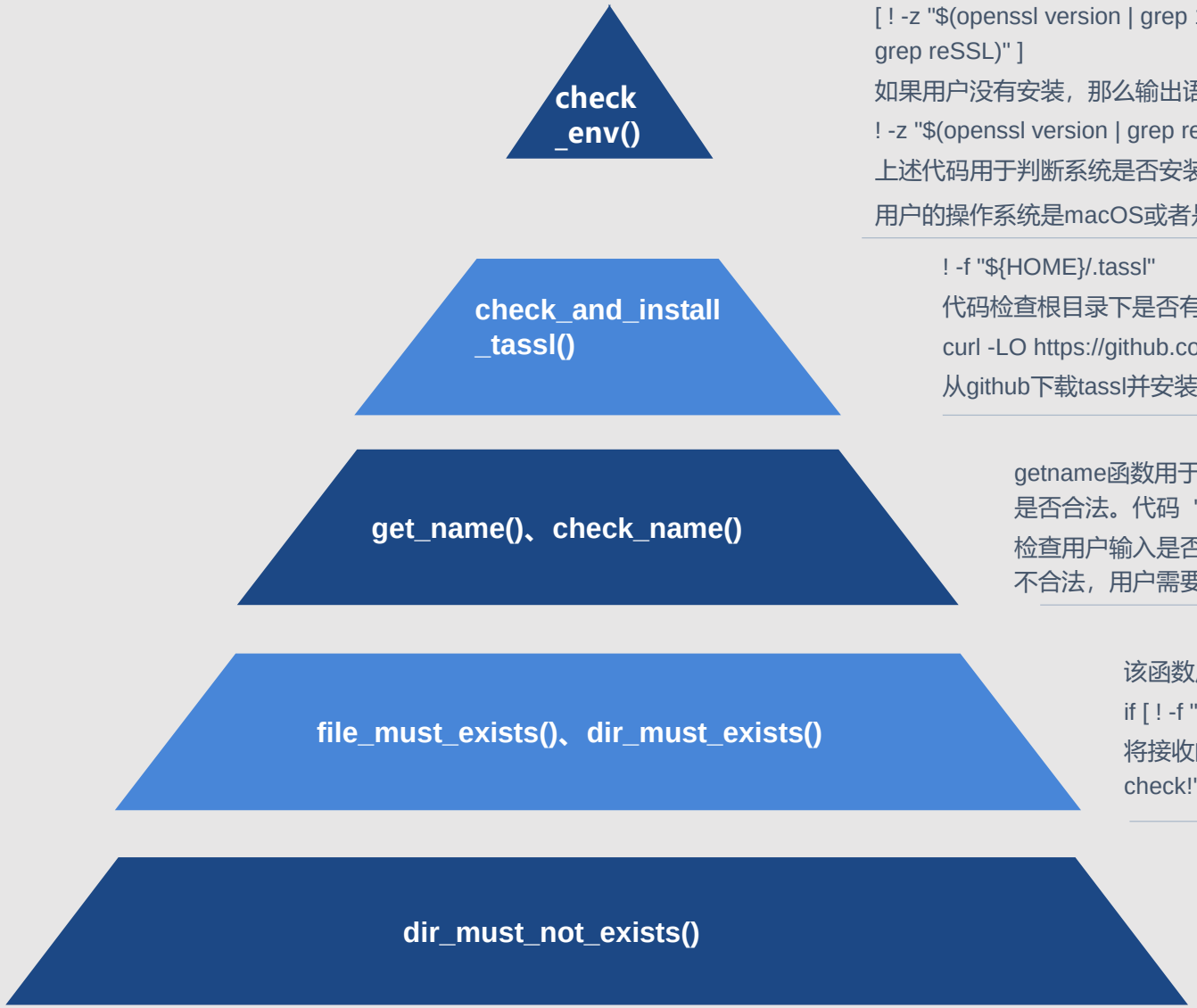
LOG_WARN()、LOG_INFO()

两个函数的作用是输出登陆警告和登陆信息。输出警告的格式是WARN+警告内容，输出登陆信息的格式是INFO+错误内容。

fail_message()

echo $1输出传入的第一个参数，即对应的失败信息。

检查系统环境、参数合法性



该函数用于检查系统的环境，由于build-chain.sh脚本依赖于openssl，所以在执行时要检查系统是否安装1.0.2或者1.1版本的openssl。

[! -z "$(openssl version | grep 1.0.2)"] || [! -z "$(openssl version | grep 1.1)"] || [! -z "$(openssl version | grep reSSL)"]

如果用户没有安装，那么输出语句提示用户下载安装。

! -z "$(openssl version | grep reSSL)"

上述代码用于判断系统是否安装reSSL，如果安装则设置路径为openssl文件夹下的bin文件夹。最后判断用户的操作系统是macOS或者是Linux，结果用变量OS记录。

! -f "${HOME}/.tassl"

代码检查根目录下是否有tassl文件，如果没有，则执行代码

curl -LO https://github.com/FISCO-BCOS/LargeFilesraw/master/tools/tassl.tar.gz

从github下载tassl并安装启动。

getname函数用于接收用户输入，并将其赋给全局变量中。check-name函数用于检查用户输入的name是否合法。代码 "$value" =~ ^[a-zA-Z0-9._-]+$

检查用户输入是否只包含英文大小写、数字、下划线等符号，如果输入不符合上述规则，则提示name不合法，用户需要遵循上述规则。

该函数用来检查目标文件是否存在，代码

if [! -f "$1"]; then echo "$1 file does not exist, please check!"

将接收的文件名传给$1，并且如果该文件不存在，将输出$1 DIR does not exist, please check!"提示用户文件不存在。dir-must-exists函数同理。

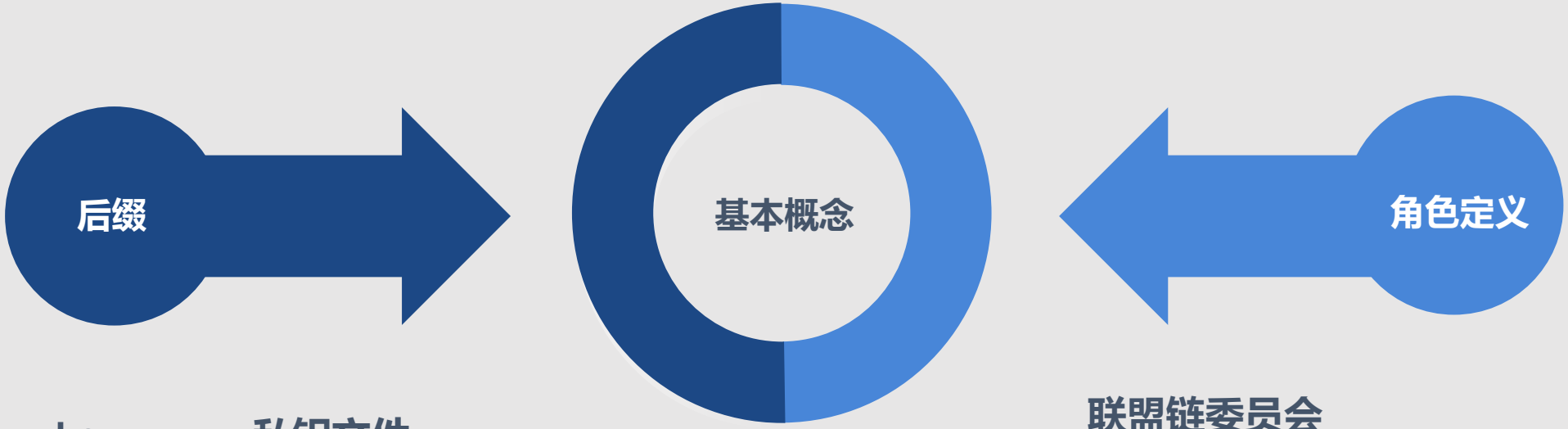
该函数用于检查目标文件夹是否已经存在，代码

if [-e "$1"]; then echo "$1 DIR exists, please clean old DIR!"

将接收到的文件夹名传给$1，如果该文件夹已经存在，则输出"$1 DIR exists, please clean old DIR!"提示用户清空原有文件夹。

2 生成证书函数

证书说明



.key	私钥文件
.crt	证书文件
.csr	证书请求文件

联盟链委员会

机构

节点/SDK

证书生成过程


机构的证书请求文件
agency.csr
节点的证书请求文件
node.csr
联盟链委员会
机构
节点/SDK
机构证书
agency.crt
节点/SDK证书
node.crt
管理链的私钥
管理机构的私钥

gen_chain_cert()

- 联盟链委员会使用openssl命令请求链私钥，并输出在*ca.key*文件中，长度为2048
- 根据ca.key生成链证书*ca.crt*
- **genrsa**为生成并输入一个rsa私钥，rsa为非对称加密算法
- **-out**为输出的路径
- 使用 **-new**，说明是要生成证书请求，当使用 **-x509**选项的时候，说明是要生成自签名证书
- **gen_chain_cert_gm()** 为使用国密版本的证书生成函数

```
gen_chain_cert() {
    path="$2"
    name=$(getname "$path")
    echo "$path --- $name"
    dir_must_not_exists "$path"
    check_name chain "$name"

    chaindir=$path
    mkdir -p $chaindir
    openssl genrsa -out $chaindir/ca.key 2048
    openssl req -new -x509 -days 3650 -subj "/CN=$name/O=fisco-bcos/OU=chain" -key $chaindir/ca.key -out $chaindir/ca.crt
    mv cert.cnf $chaindir
}
```

gen_agency_cert()

- 使用openssl命令生成机构私钥*agency.key*
- 机构使用机构私钥*agency.key*得到机构证书请求文件*agency.csr*
- 联盟链委员会使用链私钥*ca.key*，根据得到机构证书请求文件*agency.csr*生成机构证书*agency.crt*
- 机构接受证书，并对其进行整合
- **gen_agency_cert_gm()** 为国密版本的证书生成函数

```
openssl genrsa -out $agencydir/agency.key 2048
openssl req -new -sha256 -subj "/CN=$name/O=fisco-bcos/OU=agency" -key $agencydir/agency.key -config $chain/cert.cnf -out $agencydir/ag
openssl x509 -req -days 3650 -sha256 -CA $chain/ca.crt -CAkey $chain/ca.key -CAcreateserial\
    -in $agencydir/agency.csr -out $agencydir/agency.crt  -extensions v4_req -extfile $chain/cert.cnf
```

gen_node_cert()

- 节点生成私钥*node.key*和证书请求文件*node.csr*
- 机构管理员使用私钥*agency.key*和证书请求文件*node.csr*为节点颁发证书*node.crt*
- **openssl ec**为椭圆曲线密钥处理工具,**-text**:print the key;**-in**:input file;**-noout**:don't print key out
- **gen_node_cert_gm()** 为国密版本的证书生成函数

```
gen_cert_secp256k1 "$agpath" "$ndpath" "$node" node
#nodeid is pubkey
openssl ec -in $ndpath/node.key -text -noout | sed -n '7,11p' | tr -d ": \n" | awk '{print substr($0,3);}' | cat >$ndpath/node.nodeid
# openssl x509 -serial -noout -in $ndpath/node.crt | awk -F= '{print $2}' | cat >$ndpath/node.serial
cp $agpath/ca.crt $agpath/agency.crt $ndpath

cd $ndpath

echo "build $node node cert successful!"
```

generate_cert_conf()

generate_cert_conf_gm()

- 显示了一些证书的基本信息，以及一些配置的默认设置
- 默认使用***sha256***计算哈希值进行加密
- ***HOME*** 为起始的地址
- ***RANDFILE*** 为读取和写入随机数种子信息的文件
- ***oid_section*** ： 指定了一个字段，该字段配置文件中包含的额外的对象标识符
- 上面还有TSA实例所使用的Policy

```
generate_cert_conf_gm()
{
    local output=$1
    cat << EOF > ${output}
HOME                    = .
RANDFILE                = $ENV::HOME/.rnd
oid_section             = new_oids
[ new_oids ]
tsa_policy1 = 1.2.3.4.1
tsa_policy2 = 1.2.3.4.5.6
tsa_policy3 = 1.2.3.4.5.7
####################################################################
[ ca ]
default_ca      = CA_default            # The default ca section
}
```

3

Generate script

函数

generate_script_template 函数

-生成脚本的模板，即向每个生成的脚本文件中导入相同的头两行的内容

```
#!/bin/bash
SHELL_FOLDER=\$(cd \$(dirname \$0);pwd)
```

generate_node_scripts函数

-在生成的node*文件夹中，生成运行和暂停节点的脚本（start.sh/stop.sh）

gen_TransTest函数

-生成.transTest.sh脚本，该脚本是通过获取端口信息来发送交易请求验证是否可以完成交易的。

-.transTest.sh 包含获取节点版本、区块限制、发送验证信息等功能

generate_server_scripts函数

-生成server脚本，即运行和暂停所有节点的脚本（start_all.sh/stop_all.sh）

4 配置文件函数

•generate_config_ini函数

变量设置

```
local output=${1}
local ip=${2}
local offset=${ip_node_counts[${ip//./}]}
local node_groups=(${3//,/ })
local prefix=""
if [ -n "$guomi_mode" ]; then
                    prefix="gm"
```

关于基础变量

接收ip等参数，包括群节点等信息，进行文件配置。

•generate_config_ini函数

```
[rpc]
; rpc listen ip
listen_ip=${listen_ip}
; channelserver listen port
channel_listen_port=$((offset + port_start[1]))
; jsonrpc listen port
jsonrpc_listen_port=$((offset + port_start[2]))
```

设置远程过程调用服务的ip，端口等参数

其中包括监听端口，ip地址等信息

•generate_config_ini函数

p2p情况下的ip，端口配置，如上段，但这里需要采用广播地址，端口设置基本与上段类似

```
[p2p] ;
 p2p listen ip listen_ip=0.0.0.0
 ; p2p listen port
 listen_port=$((offset + port_start[0]))
; nodes to connect
$ip_list
;enable/disable network compress
;enable_compress=false
```

•generate_config_ini函数

;certificate rejected list [certificate_blacklist] ; crl.0 should be nodeid, nodeid's length is 128 ;crl.0= ;group configurations ;WARNING: group 0 is forbided。

黑名单设置

[network_security] ; directory the certificates located in data_path=${conf_path}/ ; the node private key file key=${prefix}node.key ; the node certificate file cert=${prefix}node.crt ; the ca certificate file ca_cert=${prefix}ca.crt ; storage security releated configurations [storage_security] ; enable storage_security or not ;enable=true ; the IP of key mananger ;key_manager_ip= ; the Port of key manager ;key_manager_port= ;cipher_data_key=。

设定数据路径，以及配置证书。此外，可以通过设置key manager的信息来确保储存信息的安全性。

[chain] id=${chain_id} [compatibility] supported_version=${fisco_version}。

根据此前设置的chain_id配置链

- **generate_config_ini函数**

```
[log]
 ; the directory of the log
log_path=./log
 ; info debug trace
level=${log_level}
; MB
 max_log_file_size=200
 ; control log auto_flush
flush=${auto_flush}
 ; easylog config
format=%level|%datetime{%Y-%M-%d %H:%m:%s:%g}|%msg
log_flush_threshold=100
```

配置完成后记录日志。根据指定的配置路径存放日志，并且设置了日志的最大限制，使得能够自动刷新日志。

•generate_group_genesis函数

变量设置

```
local output=$1
local index=$2
local node_list=$3
```

根据传入参数配置序号，节点等信息。

```
;consensus configuration
 [consensus]
 ;consensus algorithm type, now support PBFT(consensus_type=pbft) and Raft(consensus_type=raft)
consensus_type=${consensus_type} ;the max number of transactions of a block
max_trans_num=1000 ;the node id of leaders ${node_list}
```

初始化设定区块内交易次数最大限制，同时允许用户选择不同的算法应用，如PBFT（拜占庭容错）和Raft算法，能够很好地解决区块链中一致性问题。

•generate_group_genesis函数

[storage]
;storage db type, leveldb or external
type=${storage_type}
topic=DB
设置储存形式

•[state]
•;support mpt/storage
•type=${state_type}
•;tx gas limit
•[tx]
•gas_limit=300000000
设定state_type，并设定了最大限制数量。

[group]
id=${index}
timestamp=${timestamp}
收参数配置序号和发生的时间戳。

•generate_group_ini函数

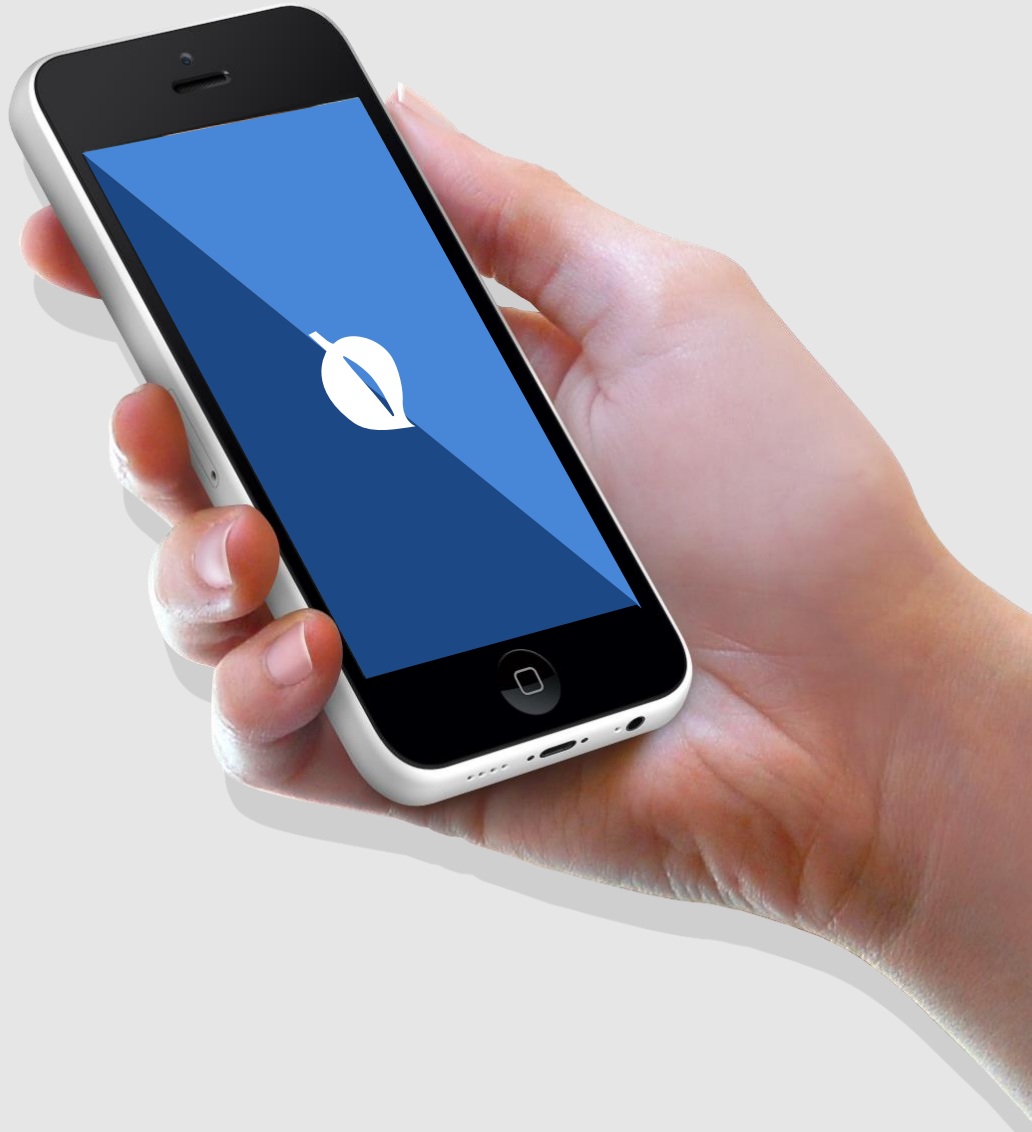
设定区块产生的时间限制，因此发生交易之后区块高度的变化受制于机器性能。

[consensus]

;ttl=2

;min block generation time(ms), the max block generation time is 1000 ms ;min_block_generation_time=500

;enable_dynamic_block_size=true



•generate_group_ini函数

设定池最大限制，并线性执行。

```
;txpool limit
[tx_pool]
limit=150000
[tx_execute]
enable_parallel=true
```

5 Main函数

Main函数

判断参数`use_ip_param`的值

- 为空，说明没用-l或-f
- 为真，说明用了-f，从命令行参数中获取配置路径的值
- 为假，说明用了-l

```
output_dir="$(pwd)/${output_dir}"
[ -z $use_ip_param ] && help 'ERROR: Please set -l or -f option.'
if [ "${use_ip_param}" == "true" ]; then
  ip_array=(${ip_param//,/ })
elif [ "${use_ip_param}" == "false" ]; then
  if ! parse_ip_config $ip_file; then
    echo "Parse $ip_file error!"
    exit 1
  fi
else
  help
fi
```

Main函数

- 断言`nodes`文件夹不存在，如果存在，则说明建链脚本已经运行过了
- 判断是否自定义 fisco_version，一般自动去获取版本，然后用sed获取版本号

```
dir_must_not_exists ${output_dir}
mkdir -p "${output_dir}"

# get fisco_version
if [ -z "${fisco_version}" ]; then
    fisco_version=$(curl -s https://raw.githubusercontent.com/FISCO-BCOS/FISCO-BCOS/master/release_note.txt | sed "s/^[vV]//")
fi
```

Main函数

在非docker模式下

判断是否自定义 fisco-bcos二进制文件的路径以及OS的类型是否已知？

- 没有自定义或不知道OS类型：判断是不是国密模式，然后到github中下载相应版本的压缩包；然后下载，解压，授权；
- 没有自定义但是OS不为空：说明是MacOS　// 这里有点没懂
- 自定义路径：
    1. 检查该路径下的二进制文件是否正确；
    2. 检查是不是国密版本；
    3. 检查输入模式和文件模式是否匹配

```
bin_version=$(${bin_path} -v)
if [ -z "$(echo ${bin_version} | grep 'FISCO-BCOS')" ]; then
  LOG_WARN "${bin_path} is wrong. Please correct it and try again."
  exit 1
fi
if [[ ! -z ${guomi_mode} && -z $(echo ${bin_version} | grep 'gm') ]]; then
  LOG_WARN "${bin_path} isn't gm version. Please correct it and try again."
  exit 1
fi
if [[ -z ${guomi_mode} && ! -z $(echo ${bin_version} | grep 'gm') ]]; then
  LOG_WARN "${bin_path} isn't standard version. Please correct it and try again."
  exit 1
fi
echo "Binary check passed."
```

Main函数

- 如果没有用-t指定证书，那就在当前目录下新建一个cert.cnf证书;
如果有指定，就拷贝过来

- 如果是通过配置文件配置参数的，执行这个——我没看懂

```
if [ "${use_ip_param}" == "true" ]; then
  for i in $(seq 0 ${#ip_array[*]}); do
    agency_array[i]="agency"
    group_array[i]=1
  done
fi
```

Main函数

生成CA证书

- 断言 /nodes/chain 不存在
- 生成chain证书在 /nodes/chain下，再拷贝至 /nodes/cert中
- 又根据配置，生成agency证书

生成国密证书

- 检查安装tassl ?
- 生成cnf，cert，key等。。。

```
# prepare CA
echo "============================================================="
if [ ! -e "$ca_file" ]; then
  echo "Generating CA key..."
  dir_must_not_exists ${output_dir}/chain
  gen_chain_cert "" ${output_dir}/chain >${output_dir}/${logfile} 2>&1 || fail_message "openssl error!"
  mv ${output_dir}/chain ${output_dir}/cert
  if [ "${use_ip_param}" == "false" ]; then
    for agency_name in ${agency_array[*]}; do
      if [ ! -d ${output_dir}/cert/${agency_name} ]; then
        gen_agency_cert "" ${output_dir}/cert ${output_dir}/cert/${agency_name} >${output_dir}/${logfile} 2>&1
      fi
    done
  else
    gen_agency_cert "" ${output_dir}/cert ${output_dir}/cert/agency >${output_dir}/${logfile} 2>&1
  fi
  ca_file="${output_dir}/cert/ca.key"
fi
```

Main函数

给所有ip生成证书私钥之类的

- 对每个ip，格式`127.0.0.1:4`，获取ip和节点数num
- 迭代节点数num，生成节点目录node0， node1...
- 然后对每个节点生成自己的节点证书等。。如图一
- 如果是国密模式，进行一通操作。。如图二
- 然后对每个ip生成sdk，似乎也是一堆证书

```
3
4   echo "============================================================="
5   echo "Generating keys ..."
6   nodeid_list=""
7   ip_list=""
8   count=0
9   server_count=0
0   groups=
1   ip_node_counts=
2   groups_count=
3   for line in ${ip_array[*]}; do…
9   done
```

循环

Main函数

每个节点生成
自己的节点证书

```
while :; do
  gen_node_cert "" ${output_dir}/cert/${agency_array[${server_count}]} ${node_dir} >${output_dir}/${logfile} 2>&1
  mkdir -p ${conf_path}/
  rm node.param node.private node.pubkey agency.crt
  mv *.* ${conf_path}/

  #private key should not start with 00
  cd ${output_dir}
  privateKey=$(openssl ec -in "${node_dir}/${conf_path}/node.key" -text 2>/dev/null | sed -n '3,5p' | sed 's/://g' | t
  len=${#privateKey}
  head2=${privateKey:0:2}
  if [ "64" != "${len}" ] || [ "00" == "$head2" ]; then
    rm -rf ${node_dir}
    continue
  fi

  if [ -n "$guomi_mode" ]; then
    gen_node_cert_gm "" ${output_dir}/gmcert/agency ${node_dir} >${output_dir}/build.log 2>&1
    mkdir -p ${gm_conf_path}/
    mv ./*.* ${gm_conf_path}/

    #private key should not start with 00
    cd ${output_dir}
    privateKey=$($TASSL_CMD ec -in "${node_dir}/${gm_conf_path}/gmnode.key" -text 2>/dev/null | sed -n '3,5p' | sed 's
    len=${#privateKey}
    head2=${privateKey:0:2}
    if [ "64" != "${len}" ] || [ "00" == "$head2" ]; then
      rm -rf ${node_dir}
      continue
    fi
  fi
  break
done
```

图一　　//整不明白

Main函数

国密模式下的一些操作

```
if [ -n "$guomi_mode" ]; then
  cat ${output_dir}/gmcert/agency/gmagency.crt >>${node_dir}/${gm_conf_path}/gmnode.crt
  cat ${output_dir}/gmcert/gmca.crt >>${node_dir}/${gm_conf_path}/gmnode.crt

  #move origin conf to gm conf
  rm ${node_dir}/${conf_path}/node.nodeid
  cp ${node_dir}/${conf_path} ${node_dir}/${gm_conf_path}/origin_cert -r
fi

if [ -n "$guomi_mode" ]; then
  nodeid=$($TASSL_CMD ec -in "${node_dir}/${gm_conf_path}/gmnode.key" -text 2>/dev/null | pe
else
  nodeid=$(openssl ec -in "${node_dir}/${conf_path}/node.key" -text 2>/dev/null | perl -ne '
fi

if [ -n "$guomi_mode" ]; then
  #remove original cert files
  rm ${node_dir:?}/${conf_path} -rf
  mv ${node_dir}/${gm_conf_path} ${node_dir}/${conf_path}
fi
```

图二

Main函数

- 给所有ip生成对应的节点配置和启动停止脚本
- 最后删掉log信息，打印建链结果

```
ip_node_counts=()
echo "============================================================="
echo "Generating configurations..."
cd ${current_dir}
server_count=0
for line in ${ip_array[*]}; do…
done

rm ${output_dir}/${logfile}
if [ "${use_ip_param}" == "false" ]; then
  echo "============================================================="
  for l in $(seq 0 ${#groups_count[@]}); do
    if [ ! -z "${groups_count[${l}]}" ]; then echo "Group:${l} has ${groups_count[${l}]} nodes"; fi
  done
fi
```

循环