I finally made it, if it can help someone. I don't know if it is nice, but it works nice with POSIX rights, without ugly ACL.

1] Create groups

```
$ sudo groupadd Group1
$ sudo groupadd ...
```

Please notice I did not create an Admin group since my admin users will be in all the other groups. This is the only way I made it work so far. If someone has a nicer way to deal with for instance an admin group with rights on anything, please tell me ! My solution is correct since I am managing a dozen of accounts but not hundreds or thousands...

2] Create users (without home folder and ability to use SSH for security reasons)

```
$ sudo adduser --no-create-hom --shell /usr/sbin/nologin user1
$ sudo adduser --no-create-hom --shell /usr/sbin/nologin user...
```

3] Add users in smbpasswd database. Please pay attention to the fact I had to add them, but also to enable them to avoid any kind of problems.

```
$ sudo smbpasswd -a user1
$ sudo smbpasswd -e user1
$ ...
```

4] Add users to the groups where they will be. Please notice I thought samba was not able to recognize secondary groups, but actually it does. The mistake was a misusing of the command line.

The error was:

```
$ sudo usermod -G user1 Group1
$ sudo usermod -G user1 Group2
```

If you do:

```
$ groups user1
```

You have something like:

```
user1 Group2
```

Because the secondary group has disappeared.

The good practice is:

```
$ sudo usermod -G Group1,Group2,Group3 user1
```
In this case:

```
$ groups user1
```
Will show:

```
$ user1 Admin Group1 Group2 Group3
```
Please, be careful of primary and secondary groups. An user can be in multiple secondary groups, but only in one primary group.

So another mistake would be:

```
$ sudo usermod -g Group1
```
If you do:

```
$ groups user1
```
You got:

```
$ Group1
```
4] Create the folders with the good permissions:

```
$ sudo mkdir /path/to/Directory1
$ sudo chown root:Group1 /path/to/Directory1
$ sudo chmod 2775 /path/to/Directory1
```
Please read this if you don't know nothing about file system permissions and setuid, setgid : chmod + setuid and setgid. The permissions and also there inheritances are important for your groups.

5] Create as many directories you want with the permissions you want.

6] Configure your smb.conf file with the editor of your choice. I personally use vim. Make a backup first before losing anything that was running well.

```
$ sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.bak
$ sudo vim /etc/samba/smb.conf
```
Your config file here, with global config and Directory1 config is:

```
[global]

      workgroup = YOUR-LOCAL-WORKGROUP
      server string = %h # hostname
      log file = /var/samba/smb-%h.log
```

```
        max log size = 1000
        disable netbios = yes # since this is a standalone server
        server role = standalone server
        veto files = /*.exe/*.com/*.dll/*.bat/*.vbs/*.tmp/ # whatever your want
        delete veto files = yes

[Directory1]

        path = /path/to/Directory1
        browseable = yes
        guest only = no
        guest ok = yes
        read list = nobody guest
        write list = @Group1
        force create mode = 0665 # please see system file permissions ...
        force directory mode = 2775 # ... and setuid, setgid, right above !
```

Please note the execution is important for your users. This allows them to navigate through the directories. But the execution is also a danger regarding files. You don't want you users to execute scripts or whatsoever, reading is widely enough.

That is why I restrict files on reading and writing for users and reading for anonymous, while I let executing on directories for navigation. Execute a folder is not a danger but execute a file is.

```
      force create mode = 0665
      force directory mode = 2775
```

Note the options `delete veto file` and `veto files` are also another shield since you don't want your silly Windows Mac OS X users put blobs or binaries or executable randomwares on your clean server.

7] Test your config file with a gentle command:

```
$ testparm
```

8] Everything is good ? Then.

```
$ sudo systemctl start smbd.service # or restart
$ sudo systemctl enable smbd.service
```

Now you can log with the method of your choice and see that anonymous can explore the files on the server without the permission of writing it while your authenticated and designated users can read, write and execute -- i.e navigate from a directory to a directory.

Et voilà ! I hope it will help someone.