NFS-RODS: A Tool for Accessing iRODS via the NFS Protocol

Presenter: Danilo Oliveira - dmo4@cin.ufpe.br

D.Oliveira, I. Fé, A. Lobo Jr., F. Silva, G. Callou, V. Alves, P. Maciel

Center for Informatics UFPE, Brazil

Stephen Worth

EMC Corporation

Jason Coposky

iRODS Consortium

Agenda

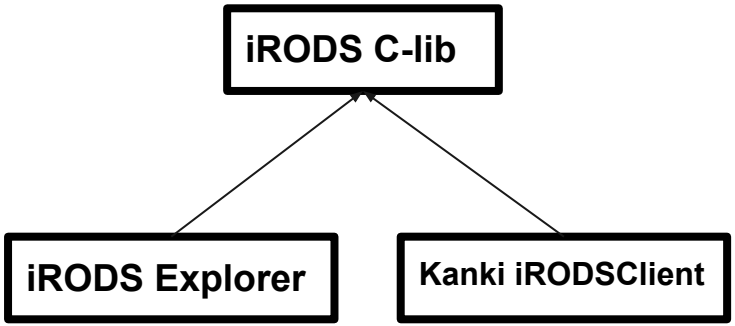
- Introduction
- NFS overview
- NFS-RODS
- Final Remarks
- (Live demonstration)

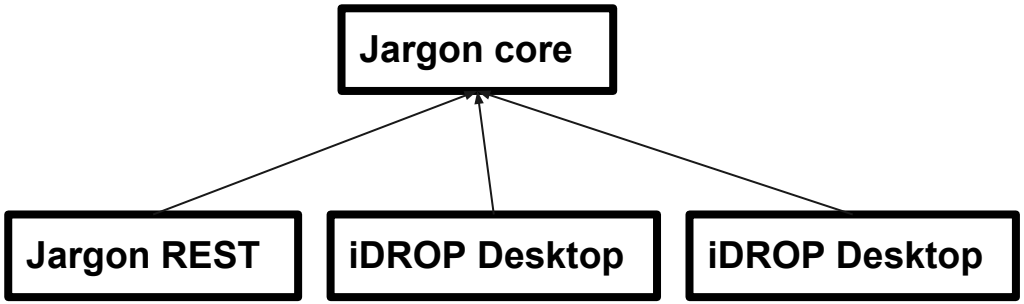
Introduction

- iRODS is a powerful data grid middleware, plenty of useful features on server side
- However, in order to be useful for final users, it depends on our ability to create client applications that address user’s needs

Introduction

Many ways for accessing iRODS...





Introduction

How can we access iRODS collections as local folders transparently (hiding the details about iRODS from the clients)?

Introduction

The NFS-RODS project aims to deliver access to an iRODS environment via typical NFS clients.

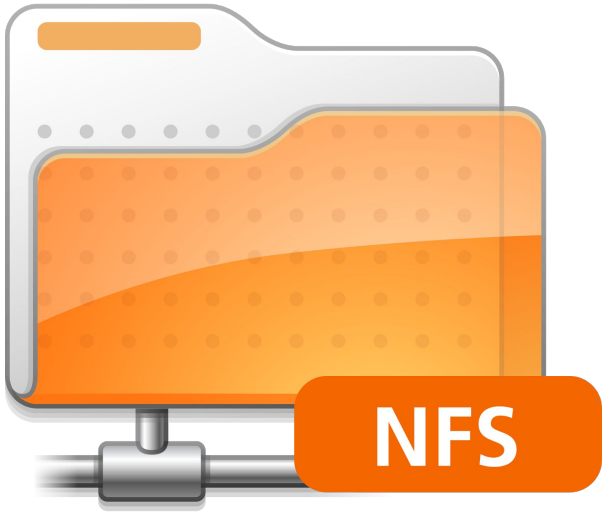
This project was created on top of UNFS (user space NFS server), and the iRODS C API

UNFS home page: http://unfs3.sourceforge.net/

NFS

Network File System (NFS) is a protocol that defines a distributed file system.

The main goal of the NFS protocol is to turn the remote file access transparent for the O.S. users.



NFS

- Technical details:
  - Transport layer: could be UDP or TCP
  - Session layer: Remote Procedure Call
  - Presentation layer: External Data Representation (XDR) protocol (also developed by Sun Microsystems)
  - Stateless server: the server doesn’t maintain information between successive client requests

NFS-RODS

NFS and iRODS are similar in some aspects, but, different in others.

The main challenge of this project was how to map the functions specified by the NFS v3 RFC to iRODS calls.

GETATTR: Get file attributes
SETATTR: Set file attributes
LOOKUP: Lookup filename
ACCESS: Check access permission
READ: Read from file
WRITE: Write to file
CREATE: Create a file
MKDIR: Create a directory
READDIR: Read From directory
REMOVE: Remove a file
RMDIR: Remove a directory
RENAME: Rename a file or directory

LINK: Create link to an object
READDIRPLUS: Extended read from directory
FSSTAT: Get dynamic file system information
FSINFO: Get static file system information
PATHCONF: Retrieve POSIX information
COMMIT: Commit cached data on a server to stable storage
SYMLINK: Create a symbolic link
MKNOD: Create a special device
READLINK: Read from symbolic link

NFSv3 procedures

Non-correspondence of attributes

NFS

```
ftype3 type;
uint32 nlink;
size3 size;
size3 used;
specdata3 rdev;
uint64 fsid;
fileid3 fileid;
nfstime3 atime;
nfstime3 mtime;
nfstime3 ctime;
mode3 mode;
uid3 uid;
gid3 gid;
```

iRODS

```
#define COL_D_DATA_ID 401
#define COL_D_COLL_ID 402
#define COL_DATA_NAME 403
#define COL_DATA_REPL_NUM 404
#define COL_DATA_VERSION 405
#define COL_DATA_TYPE_NAME 406
#define COL_DATA_SIZE 407
#define COL_D_RESC_NAME 409
#define COL_D_DATA_PATH 410
#define COL_D_OWNER_NAME 411
#define COL_D_OWNER_ZONE 412
#define COL_D_REPL_STATUS 413
#define COL_D_DATA_STATUS 414
#define COL_D_DATA_CHECKSUM 415
```

[...]

Non-correspondence of attributes

NFS

atime -> access
mtime -> modify content
ctime -> change attributes

iRODS

Same value

COL_D_CREATE_TIME -> creation

COL_D_MODIFY_TIME -> change attributes

Non-correspondence of attributes

Our solution:
copy the same value
for all fields

NFS

iRODS

Same value

atime -> access
mtime -> modify content
ctime -> change attributes

COL_D_CREATE_TIME -> creation

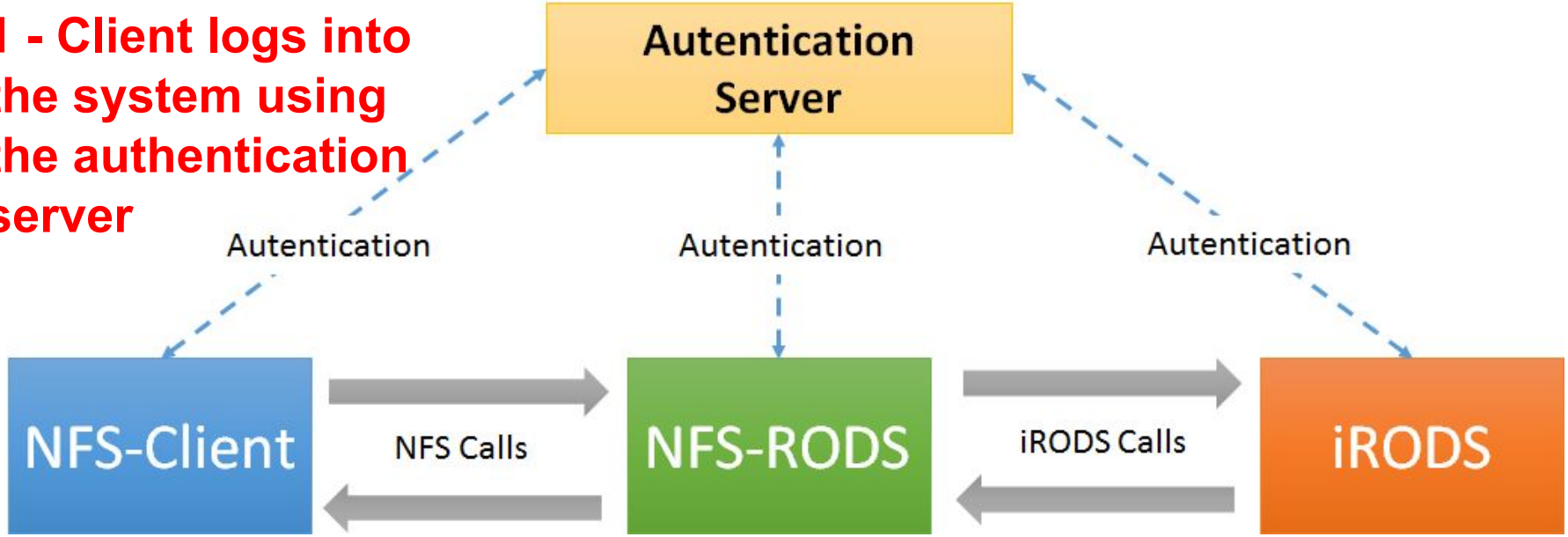
COL_D_MODIFY_TIME -> change attributes

Change Permissions

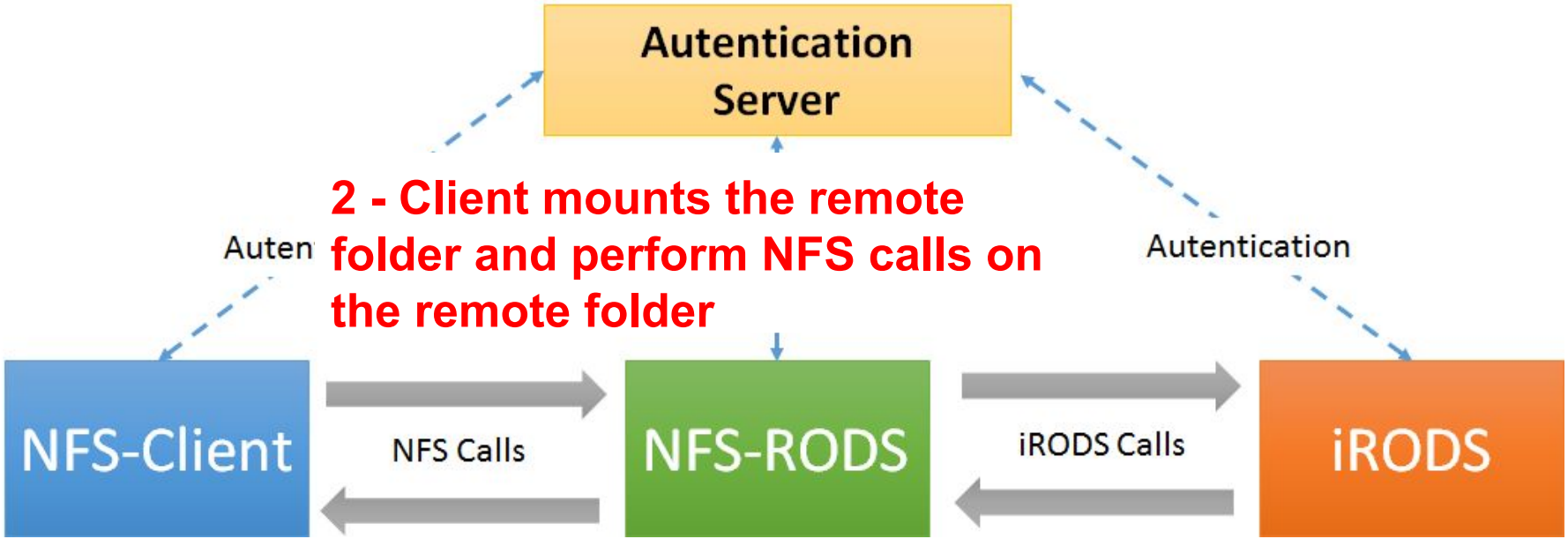
Most permissive mode unix	iRODS
chmod 7xx	own
chmod 6xx	own
chmod 5xx	Read
chmod 4xx	Read
chmod 3xx	Write
chmod 2xx	Write
chmod 1xx	Read
chmod 0xx	Null

System Architecture

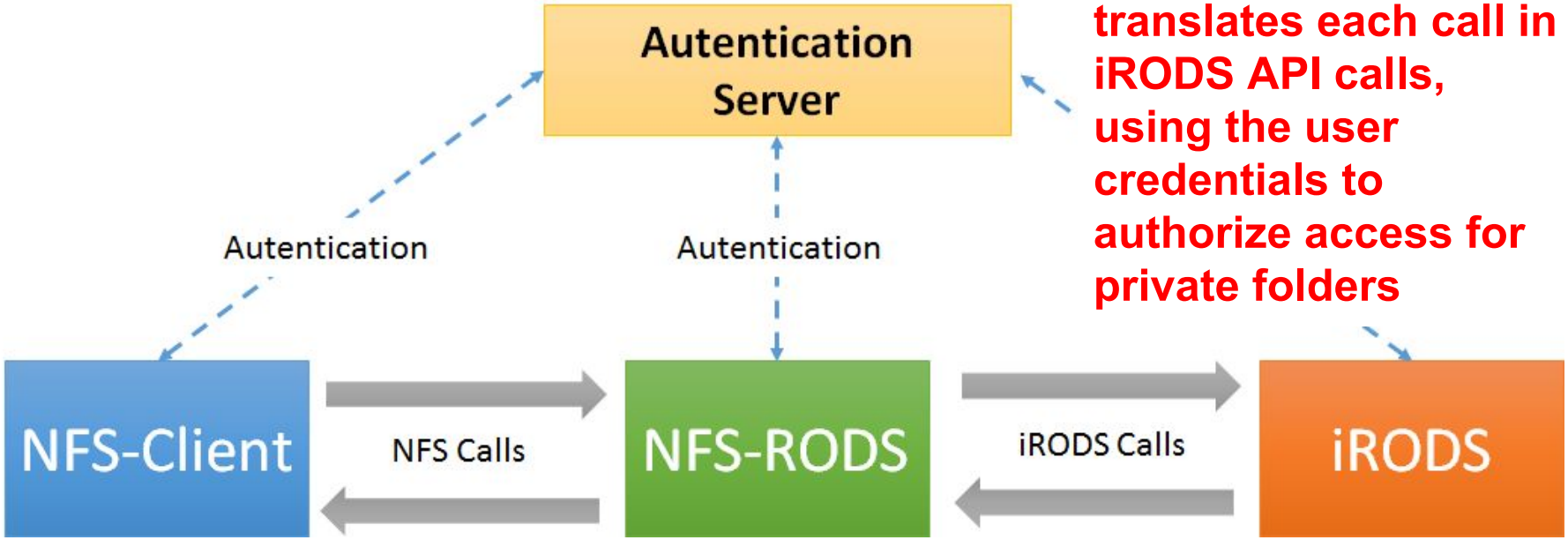
1 - Client logs into the system using the authentication server



System Architecture



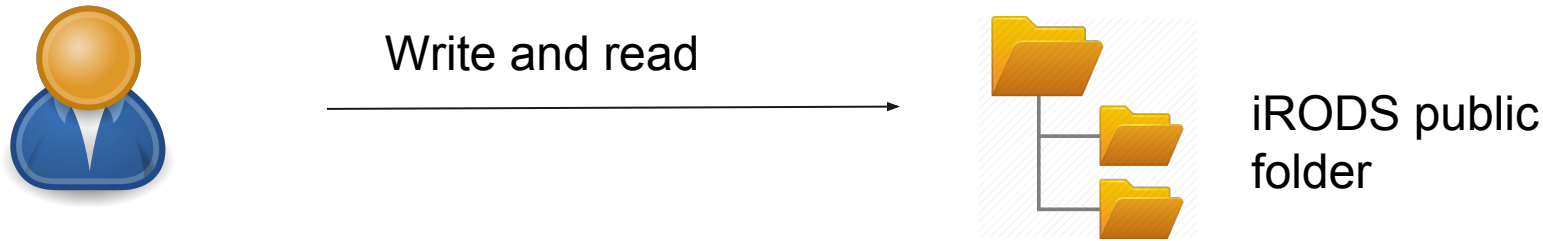
System Architecture



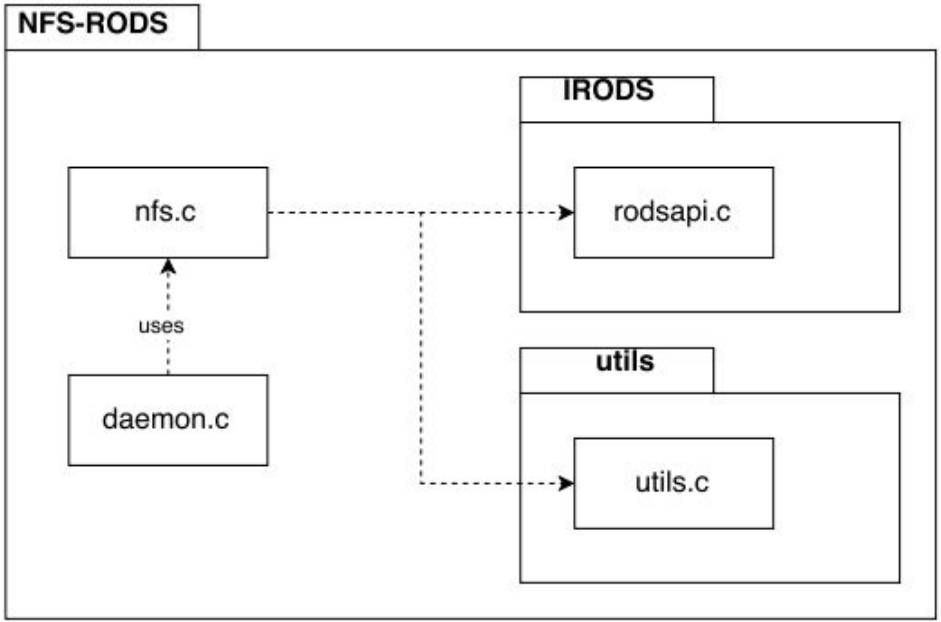
3 - NFSRODS server translates each call in iRODS API calls, using the user credentials to authorize access for private folders

System Architecture

- The authentication server is not mandatory, if we want to access folders inside the `/<zone name>/home/public` folder



Package Diagram



Installing and running NFS-RODS

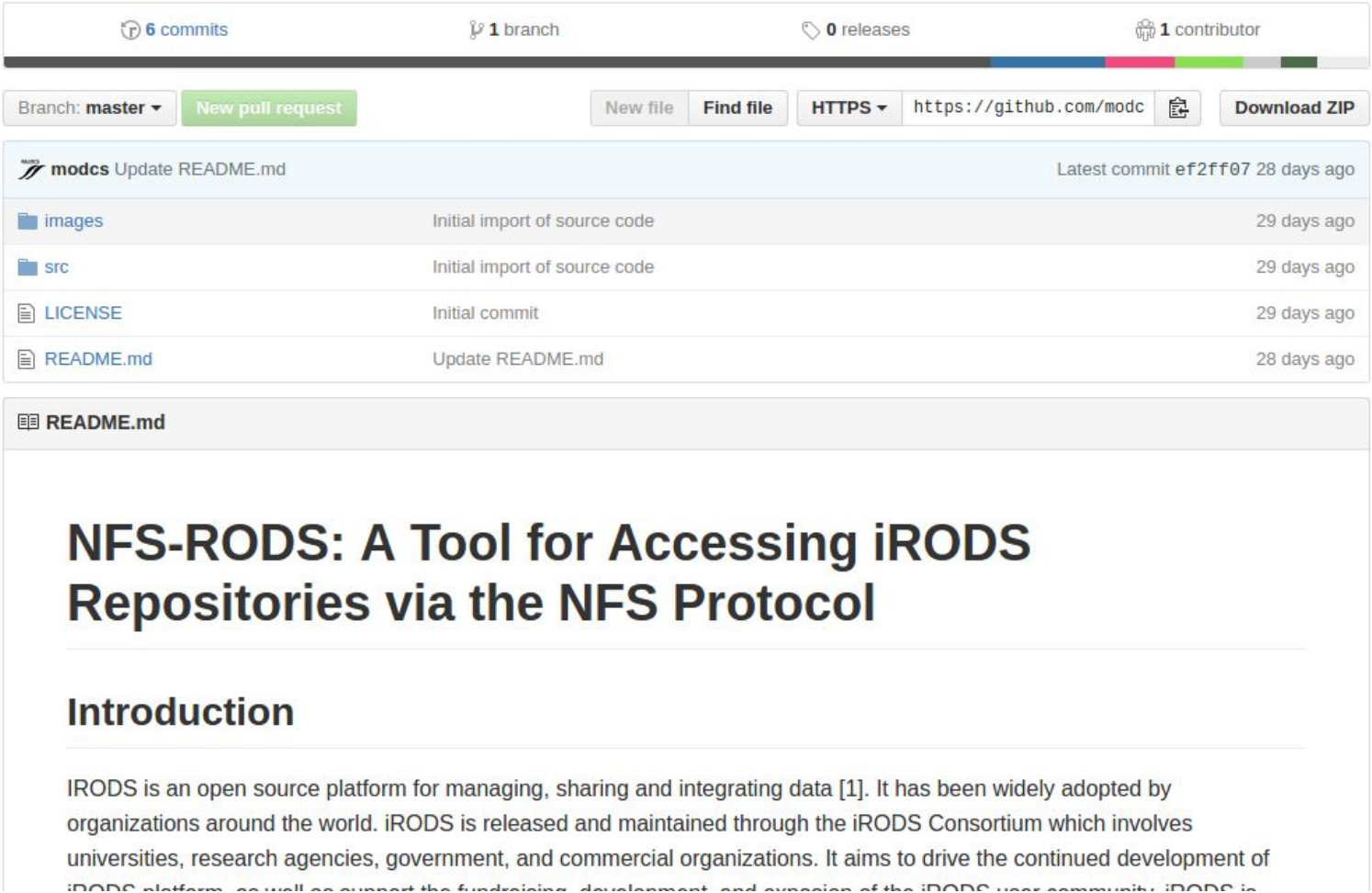
1. Clone the git repository:

   ```
   git clone https://github.com/modcs/NFSRODS.git
   ```

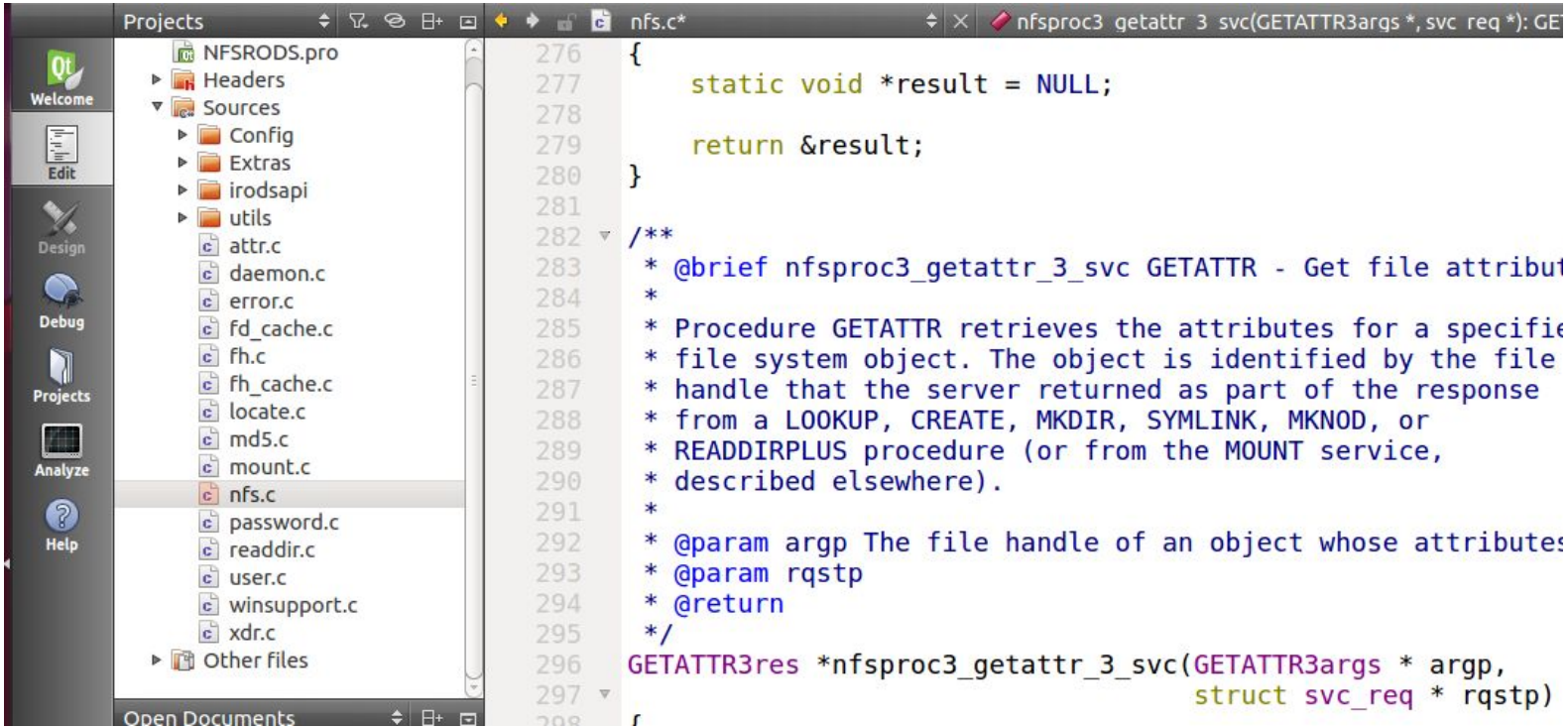
2. Run “make” on the project folder
3. Install rpcbind

   ```
   sudo apt-get install rpcbind
   ```

4. Run “./NFSRODS” (runs in background mode)


6 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
New file
Find file
HTTPS
https://github.com/modc
Download ZIP
modcs Update README.md
Latest commit ef2ff07 28 days ago
images
Initial import of source code
29 days ago
src
Initial import of source code
29 days ago
LICENSE
Initial commit
29 days ago
README.md
Update README.md
28 days ago
README.md
NFS-RODS: A Tool for Accessing iRODS Repositories via the NFS Protocol
Introduction
IRODS is an open source platform for managing, sharing and integrating data [1]. It has been widely adopted by organizations around the world. iRODS is released and maintained through the iRODS Consortium which involves universities, research agencies, government, and commercial organizations. It aims to drive the continued development of

There is a “.pro” file to open the project in QT-Creator
We are using QT-Creator as IDE, but we don’t use any QT function



Final remarks

- NFS-RODS allows administrators and users familiar with NFS to interact with iRODS
- iRODS and NFS are not fully compatible, so we had to make some decisions to integrate them in a meaningful way

Live demonstration

Any questions?

Thanks!