The iRODS CLI we deserve

Derek Dong
Software Developer
iRODS Consortium

June 13-16, 2023
iRODS User Group Meeting 2023
Chapel Hill, NC

Overview

- What are iCommands, the current default CLI?
- Why is a new CLI necessary?
- Goals for the new CLI
- Progress on the new CLI
- Remaining work
- Future

What are iCommands?

- The traditional client program that interacts with iRODS
- Comprised of 50~ or so individual binaries, each with differing functionality
- Written originally in C, then ported to C++
- A polished product: iterated upon for a long time

Why is a new CLI necessary?

- Not a single, cohesive product
- Mostly C disguised as C++
- Unfriendly to developers and newbies
- Potential breaking changes for others' workflow
- Hard dependencies on main repo

```
$ for i in $(compgen -G '*' | grep '\.cpp$' | sed 's/^i\(.*\)\.cpp/\1/')
  do
    ls ../../irods/lib/core/src/${i}Util* 2>/dev/null
  done | wc -l
  19
```

Goals for the new CLI

- Easy to comprehend and extend
- Replace/extend the original iCommands
- Use modern C++ libs (both iRODS and external)
- Easy to deploy

Progress on the new CLI - **Comprehensibility**

- Uses more boost/iRODS libs
  - Most functionality has a LoC reduction
  - Code based on the iRODS C++ libs is safer
- Avoids "hard" main repo dependencies
  - Uses the client libs, not each commands' special ones!

Progress on the new CLI - **Comprehensibility**

- Modern invocation with single base binary
- Inspired by commands like docker, ip, etc.

```
$ ./irods ls
C- '"'"'
C- potato
C- test3
bettername
testfilethree
testfiletwo
```

```
$ ./irods mv potato tomato
$ ./irods cp bettername bettername2
$ ./irods rm testfiletwo
```

Progress on the new CLI - **Comprehensibility**

```
$ ./irods ls -A
C- '"''
        ACL - rods|tempZone|own
C- test3
        ACL - rods|tempZone|own
C- tomato
        ACL - rods|tempZone|own
bettername
        ACL -     rods#own
bettername2
        ACL -     rods#own
testfilethree
        ACL -     rods#own
```

Progress on the new CLI - **Extensibility**

- Abstract class to be inherited from

```
namespace irods::cli
{
    class command
    {
    public:
        virtual ~command() = default;
        virtual auto name() const noexcept -> std::string_view = 0;
        virtual auto description() const noexcept -> std::string_view = 0;
        virtual auto help_text() const noexcept -> std::string_view = 0;
        virtual auto execute(const std::vector<std::string>& args) -> int = 0;
    };
} // namespace irods::cli
```

- Dynamically loadable modules (more later…)

Progress on the new CLI - **Feature Parity/Modernization**

Replace/extend the original iCommands

Use modern C++ libs

13~ commands with base functionality

- tree
- mkdir
- get
- put
- cd
- ls
- mv
- pwd
- touch
- etc...

All are written from scratch using modern C++ libs like irods::filesystem, libfmt, boost::program_options, etc.

Optional command linking: static or dynamic!

- Uses the power of CMake magic

Build output can be a single binary:

```
$ find . -type f \! -name '*.deb'
./irods
```

It can also be...

Progress on the new CLI - **Ease of Deployment**

```
$ find . -type f \! -name '*.deb'
./libirods_cli_put.so
./libirods_cli_mkdir.so
./libirods_cli_exit.so
./libirods_cli_get.so
./libirods_cli_cp.so
./libirods_cli_ls.so
./libirods_cli_touch.so
./libirods_cli_rm.so
./libirods_cli_tree.so
./irods
./libirods_cli_cd.so
./libirods_cli_pwd.so
./libirods_cli_mv.so
./libirods_cli_repl.so
```

Modifying linking type from static to dynamic (or vice versa?)

Just make a small edit to the CMakeLists.txt at the root of the repo!

Example of linking all subcommands statically:

```
 9 set(irods_static_subcommands ls put get repl ...)
10 set(irods_dynamic_subcommands)
```

Dynamically:

```
 9 set(irods_static_subcommands)
10 set(irods_dynamic_subcommands ls put get repl ...)
```

Demo - statically linked deployment

```
$ ls /lib/irods/plugins/cli/

$ ./irods ls
C- '"'"'
C- potato
C- test3
'
'"'"'
bettername
testfilethree
testfiletwo
```

Demo - loading plugins at runtime

```
$ ls /lib/irods/plugins/cli/

$ ./irods ls
Invalid command: ls

$ cp *.so /lib/irods/plugins/cli/

$ ./irods ls
C- '"''
C- potato
C- test3
'
'"''''
bettername
testfilethree
testfiletwo
```

Anyone building can include/exclude commands freely

```
 9 set(irods_static_subcommands ls)
10 set(irods_dynamic_subcommands put get repl ...)
```

```
$ ls /lib/irods/plugins/cli/

$ ./irods cd ..
Invalid command: cd

$ ./irods pwd
Invalid command: pwd

$ ./irods ls
C- '"''
C- test3
C- tomato
bettername
bettername2
testfilethree
```

Remaining work

- Project is young; still many commands to implement
  - Several complex ones are still pending
    - iadmin
    - imeta, etc…
- Implemented commands still missing some features
- Documentation for just about everything
  - No help messages…
- Template project for new CLI commands
- Code normalization/smell
  - Some using-namespaces are inconsistent
  - Some code is unnecessarily paranoid
- Tests of all kinds!

- Features beyond what the original iCommands have
  - JSON output on all the commands
  - Cross-platform compatibility
- Community contributions can be easily integrated, thanks to the modular system

Questions?