Advanced Unix System Administration

Lecture 5
September 29, 2008

Steven Luo
<sluo+decal@OCF.Berkeley.EDU>

Shared Libraries

- The dynamic linker
  - Binaries have a “symbol table” containing functions, etc. and their locations
  - Dynamic binaries have tables with blanks – it's the responsibility of the dynamic linker to resolve these
  - Linker loads listed dynamic libraries and tries to resolve the symbols
  - Allows shared code, but incurs a performance penalty on most architectures

Shared Libraries

- Binary compatibility
  - Programs expect the “ABI” offered by a shared library to stay the same (structs, function prototypes, etc.)
  - When this assumption breaks, things go horribly – or worse, subtly – wrong
  - Hence mechanisms for versioning shared libraries and symbols

The Unix Permissions Model

- Users and groups
  - Users and groups have numeric IDs associated with them
  - Groups can contain multiple users, or no users at all
- Process credentials
  - Each process has a set of credentials associated with it
    - Real user ID: set to the UID executing the process at the beginning of the execution
    - Real group ID

The Unix Permissions Model

- Process credentials con't
  - Effective user ID: the UID used for most permissions checks
  - Effective group ID
  - Saved set-user-ID: used for flexibility in setuid applications
  - Saved set-group-ID
  - Note that access control is always by user/group ID number!
  - Behavior can be very system-dependent – see the documentation, or try examples

The Unix Permissions Model

- File permissions
  - Files have a user/group ID associated with them
  - File permission bits: binary mask usually written as 4-digit octal
    - High digit: 1 = sticky, 2 = setgid, 4 = setuid
    - 2nd digit: 1 = user execute, 2 = user write, 4 = user read
    - 3rd digit: 1 = group execute, 2 = group write, 4 = group read
    - 4th digit: 1 = other execute, 2 = other write, 4 = other read

The Unix Permissions Model

- File permissions con't
  - Directory permissions:
    - High bit: 1 = deletion restricted, 2 = files created will have group set to directory's group
    - Execute bits mean permission to cd in
  - Access control is by the process's effective IDs
    - On Linux, there is a set of filesystem IDs, almost always equal to the effective UID
  - umask
    - Bits set in umask are masked out in permissions for newly created files

The Unix Permissions Model

- POSIX draft ACLs
  - Allow the addition of extra user and group permissions entries
  - A “mask” is set on each file and is ANDed with each ACL entry to determine effective permissions
- NFSv4 ACLs
  - Provide very granular (and different!) permissions based on a linear allow/deny list
  - More flexible, more difficult to deal with
  - Has some compatibility issues (umask, ...)