Wine/Samba

Andrew Tridgell
tridge@osdl.org

(please heckle during the talk!)

Common Technologies

- Wine and Samba face some common areas of development
  - DCE/RPC, IDL and related technologies
  - representing NTFS filesystem features on Posix
  - multiple locking APIs
- Open questions
  - What should we cooperate on?
  - should we aim for Wine/Samba interoperability?
  - if we cooperate, then how?

Different Aims

- Object level vs wire level
    - Samba doesn't care about API compatibility, Wine does
- Trust the caller?
    - for a file server, users are the enemy
    - for wine, users (applications) can be assumed benign
    - that will change when wine does networking, RPC etc

DCE/RPC and IDL

- Both projects are increasing efforts in this area
    - widl IDL compiler in wine
    - pidl IDL compiler in Samba
- Could we share IDL compiler?
    - wire compatibility versus object compatibility
    - Samba IDL extensions
- Could we share IDL files?
    - much easier!
    - Samba IDL license aims for sharing with Wine
- Share test tools?
    - rpcecho, smbtorture, ndrdump etc

DCE/RPC and IDL .... continued

- DCE/RPC test suites
  - smbtorture RPC tests
  - aiming for high coverage
  - focussing on server-oriented calls
- Samba IDL status
  - about 50% of Samba4 code generated from IDL
  - widely used for non-traditional tasks (nbt, xattrs etc)
  - 4 transports so far (ncacn_np, ncacn_ip_tcp, ncalrpc and ncacn_unix_stream)
  - good start on DCOM support

DCE/RPC differences

- Samba4 treats DCE/RPC and IDL differently from midl
    - structure based calling convention
    - all calls can be async
    - direct C (not table based) NDR handling
    - extended endpoint name syntax
    - many IDL syntax extensions
    - auto-generated size and debug functions
- Should we cooperate?
    - different goals?
    - no common IDL interfaces yet? (maybe DCOM)

NTFS features on Posix

- Both Wine and Samba need to squeeze NTFS like filesystem features on Posix
    - timestamps
    - attributes
    - OS/2 style EAs
    - streams
    - case-insensitivity
    - NT ACLs
    - 8.3 names
    - and the just plain weird stuff ....

Samba3 vs Samba4 vs Wine

- Samba3 and Samba4 approach posix mappings differently
  - Samba3 is more ad-hoc, with less clearly defined interfaces
  - Samba4 uses a 'NTVFS' interface, hiding Posix mappings behind a NT-like API
  - Samba4 aims for much more complete mapping
- Wine is more like Samba3 in approach?
  - mostly Win9x style filesystem model?
  - not aimed at fileserver oriented tasks?
  - will this change?

xattr mappings

- Samba4 uses xattrs to store most NTFS features
    - defined in xattr.idl, but not strictly tied to IDL
    - holds timestamps, dos attribs, alloc size, NT ACLs, NT streams and OS/2 style EAs
    - uses 4 separate xattr names, in 2 namespaces
        - user.DosAttrib
        - user.DosEAs
        - user.DosStreams
        - security.NTACL

File Attributes

```
const string XATTR_DOSATTRIB_NAME = "user.DosAttrib";

typedef struct {
  uint32        flags;
  uint32        attrib;
  uint32        ea_size;      /* accelerator for DosEAs */
  udlong        size;         /* used to validate alloc_size */
  udlong        alloc_size;
  NTTIME        create_time;
  NTTIME        change_time;
  NTTIME        write_time; /* only when sticky write time set */
  utf8string    name;        /* for case-insensitive speedup */
} xattr_DosInfo2;
```

OS/2 EAs

```
const string XATTR_DOSEAS_NAME = "user.DosEAs";

typedef struct {
   utf8string        name;
   DATA_BLOB value;
} xattr_EA;

typedef [public] struct {
   uint16                               num_eas;
   [size_is(num_eas)] xattr_EA *eas[];
} xattr_DosEAs;
```

NTFS named streams

```
const string XATTR_DOSSTREAMS_NAME = "user.DosStreams";

typedef struct {
  uint32     flags;
  udlong     size;
  udlong     alloc_size;
  utf8string name;
} xattr_DosStream;

typedef [public] struct {
  uint32                        num_streams;
  [size_is(num_streams)] xattr_DosStream *streams[];
} xattr_DosStreams;
```

NT ACLs

```
const string XATTR_NTACL_NAME = "security.NTACL";

typedef [switch_type(uint16)] union {
   [case(1)] security_descriptor *sd;
} xattr_NTACL_Info;

typedef [public] struct {
  uint16                  version;
  [switch_is(version)] xattr_NTACL_Info info;
} xattr_NTACL;
```

xattr tradeoffs

- The good ...
  - convenient, keeps data with file
  - available on most modern filesystems
- The bad ...
  - not enabled on many systems
  - can slow down some filesystems a lot
- The ugly
  - very small limits for streams and EAs
  - incorrect atomic semantics
  - all-at-once access only

some alternatives

- tdb backend
    - portable, supports large streams and EAs
    - not scalable to large systems. Could be split up?
    - not journaled yet
- dot-files?
    - horrible rename, unlink semantics
    - directly visible to Posix applications

LSM module

- Can we make xattrs atomic? secure?
  - yes, via a LSM module
  - speed gain via in-kernel attribute cache
  - especially useful for NT ACLs
- all the right hooks
  - LSM can give us transparent visibility of NTFS attributes to Posix (eg. ACLs)
  - need a user/kernel channel for credentials and attribute changes

case-insensitivity

- current solution
  - scan directories a lot
  - hardest problem is proving a file does not exist
  - horrible performance in some cases
- kernel support?
  - lots of resistance from kernel developers for case-insensitive filesystem support
  - maybe a coherence hook could be added?
- other methods?
  - yes, but they get complex

case-insensitivity continued ...

- possible solution?
    - store case-preserved name in xattr
    - Wine and Samba return case-preserved name
    - store lowercase name in posix directory
    - keep a shared-memory store of directory state
        - default is unknown state
        - other states based on combinations of uppercase chars to scan
    - update directory state during directory scan
    - coherence using directory timestamp or kernel seqnum?

case-insensitivity continued ...

- properties
  - Windows clients see correct semantics for windows files, plus fast access
  - posix clients see correct semantics for posix files
  - posix clients see windows files as lowercase (unless libc updated)
  - memory usage proportional to number of directories, not number of files
- coherence?
  - timestamps not ideal
  - a directory sequence number could be provided by kernel?

locking

- three flavours
    - byte range locks
    - share modes
    - oplocks
- Share between Samba and Wine?
    - Samba uses tdb databases - could be shared
    - does Wine care about exact locking semantics?
    - requires quite intimate communication

Where to now?

- Should we interoperate?
  - Wine and Samba have quite different needs, but also significant overlap
  - most obvious area for cooperation is in filesystem attributes
- Cooperate on test tools?
  - we have no automated way of running win32 tests, you do!
  - could Wine use some of our dual-server techniques? Could ReactOS?

Demos!

- smbtorture RPC-ECHO and win32 echo
- masktest
    - testing windows wildcard matching
- gentest
    - generic dual server randomised testing
- locktest
    - randomised lock testing
- file streams, ACLs
- RPC calls - such as RPC-SRVSVC