Mogua: 오픈 소스 윈도우즈

방준영
<junyoung@mogua.com>

Mogua Project

차례

- 오픈 소스 공동체의 현주소
- 프로젝트 사례
  - Wine
  - ReactOS
  - X
- Mogua 개발의 여러 가지 측면

오픈 소스 운영체제의 발전

- 데스크탑 환경
- 응용 프로그램
- 커널

문제는...

- 통일된 데스크탑 환경/API 부재
  - 사용자에게는 혼란/개발자에게는 비용 부담
  - 더 많은 시스템 자원 필요
- 호환성 부족
  - Windows 응용 프로그램 사용 불가능
  - 최신 하드웨어 미지원
- 한글 문제
  - 응용프로그램들의 국제화/지역화 미흡
  - 낮은 글꼴 품질

결과는...

- 여전히 낮은 데스크탑 시장 점유율
  - 장래성/시장성 아직 불투명
  - 언론 보도와 달리 실제적으로 Windows의 경쟁 상대가 되지 못하고 있음
- 국내 현실
  - 리눅스/BSD 사용자 공동체 활동 미약
  - 사용자 부족 -> 개발자 부족 -> 프로그램 부족 -> 사용자 부족의 악순환

역사적 사실

- 역사상 마이크로소프트의 가장 강력한 경쟁자는?
  - Unix (X), Mac OS (X), Linux (X)
  - DR-DOS (O)
- 현재 인텔의 가장 강력한 경쟁자는?
  - Alpha (X), PowerPC (X), UltraSPARC (X)
  - Athlon (O)
- 공통점: 시장 지배 제품과의 호환성

Windows 호환성

- Windows는 전세계 "사실상" 표준 운영체제
- Win32 API
  - 생각보다 그렇게 "나쁘지 않음" :-)
  - 여러 면에서 Unix API보다 우수
- 단점
  - 마이크로소프트의 시장 주도에 끌려가기 쉬움
  - 오픈소스에 덜 친화적

오픈 소스 프로젝트 사례

- Wine
- ReactOS
- X

Wine

- 1993년 Windows 3.1 프로그램을 리눅스에서 실행하기 위해 시작
- "WINdows Emulator"/"Wine Is Not an Emulator"
- 현재 리눅스, FreeBSD, NetBSD, 솔라리스에 이식되어 있음
- 전세계 수백명의 개발자들이 참여
- BSD 라이센스였으나 근래 LGPL로 전환

Wine (2)

- 기본 구조
  - Win16 NE/Win32 PE 바이너리 에뮬레이터
  - Win32 API 호환 공유 라이브러리 (libwine, libkernel32, libuser32, ...)
- Win32 API 호출을 대응하는 Unix/X API 호출로 변환하여 실행

Wine의 문제점

- 전기능을 사용자영역에서 구현
  - 커널과 사용자영역간 별도의 서버 계층 필요
  - 호환성/안정성/성능 저하
- API 변환 과정중 성능 저하/비호환성 초래
  - 이중 의존성 문제 예) NPTL, glibc와의 충돌
- 프로젝트 관리상의 문제
  - 임시방편적인 구현으로 인해 대대적인 코드 변경이 잦음

ReactOS

- 1998년 독자적인 Windows 호환 운영체제를 목표로 시작
- Windows의 기본 구조를 그대로 따름
  - 부트로더, NTOSKRNL, NTDLL, 윈도 시스템, ...
- 현재 몇몇 명령행/GUI 프로그램이 동작
  - GCC, mc, WinHello, ...
- 십여 명의 개발자들이 주축
- GPL

ReactOS의 문제점

- "바퀴를 재발명하기"
  - 특히 커널 개발은 상당히 까다롭고 시간이 오래 걸리는 작업
  - 기능 추가보다 안정성 확보 및 성능 개선이 더 어려움
- 커널 이식성 부족
- 일부 코드를 Wine에 의존

X 윈도우 시스템

- 표준 GUI 툴키트/API 미제공
  - 서로 다른 API 난립
- X 프로토콜 자체는 확장 가능
  - 그러나 X 서버는 매우 복잡
  - 실제적으로 확장하기가 어려움
- 네트워크 투명성의 효용 의문시
- 기반 OS와 기능 중첩
- XFree86의 문제

Mogua

- 2001년 이후 시작 (정확한 날짜는...?)
- 두명의 개발자
- 프로젝트 초기 단계
- 오픈 소스 (BSD 및 GPL 호환)

"바퀴 재발명하기" 문제

- 프로젝트의 처음 5년간을 커널 만드는 데 소비할 수 있음 (다음 5년은 버그 고치는 데... :-)
- 가능한 한 기존 코드를 최대한 활용하는 것이 관건
- NetBSD 커널에 기반함으로써 문제 상당 부분 해결
  - 산업계에 널리 입증된 성능과 안정성 확보
  - 깨끗한 구조와 설계를 계승

"에뮬레이트할 것인가 말 것인가" 문제

- 에뮬레이션 수준
  - 사용자 수준: Win32 API, 바이너리 변환
  - 커널 수준: 시스템 서비스, 바이너리 실행
  - 기계 수준: 가상화, 기계어 에뮬레이션
- 커널 수준 API/ABI 에뮬레이션 채택
  - 커널내 PE 바이너리 로더
  - WinNT 호환 시스템 서비스
- Windows 툴체인을 그대로 사용
  - 프로그램 재컴파일이 근본적으로 불필요

바이너리 형식

- Unix 바이너리 형식의 발전 단계
  - a.out(V7) -> COFF(SVR3) -> ELF(SVR4)
- Executable and Linking Format
  - 현재 대부분의 Unix들(리눅스, 솔라리스, *BSD 등)이 사용중인 형식
  - 이전 COFF 형식 대체
- Portable Executable
  - COFF를 확장한 마이크로소프트의 독자적 형식
  - 이전 NE 형식 대체

바이너리 형식 (2)

- ELF가 더 많은 경우에 우수
  - 구조
  - 확장성
  - 이식성
- PE는 성능면에서 우수(할 수 있음)
  - 레지스터 활용
  - 프리바인딩
- 결론: 바이너리 형식의 차이는 그렇게 크지 않음

바이너리 실행 모델

- 전통적 Unix 프로세스 모델의 문제
  - 프로세스 모델은 Windows와 잘 맞지 않음
  - 스레드 기반 모델로 이전 필요
- 오픈 소스 Unix 스레드 모델
  - POSIX Threads ("pthreads")
  - 리눅스 `clone(2)`
  - NetBSD 스케줄러 액티베이션 (SA)
  - FreeBSD 커널 스케줄러 엔티티 (KSE)
- Windows를 염두에 둔 커널 스레드 시스템 필요

스레드

- 스레드 지원의 요구 사항
  - Windows와 동일한 1:1 모델이어야
  - 스레드별 저장공간 (TLS) 지원
- 1:1 스레드 모델
  - 괜찮은 성능
  - 상대적으로 구현이 쉬움
- 스레드 레지스터
  - 스레드별 저장공간을 관리하기 위해 필요
  - i386 아키텍쳐에서는 %fs 레지스터를 이용

커널 서브시스템

- NetBSD와의 차이
  - fork() 대 CreateProcess()
  - 스레드 아이디가 시스템내 유일
  - 스케줄러에서 스레드 레지스터 조작 필요
- 가상 메모리/파일매핑 API
- 오브젝트 및 핸들 테이블
- 향후 고려 사항
  - 보안
  - 비동기 입출력

리버스 엔지니어링 (1)

- 비공개 구조를 알아내기 위해 사용
  - PEBrowse Interactive, WinDbg
  - Sang Cho's Win32 Disassembler, GNU objdump
- "Inside Windows 2000"
- "Undocumented Windows 2000 Secrets"
- "Windows NT/2000 Native API Reference"
- "Under the Hood" 컬럼, MSDN Magazine
- www.sysinternals.com

리버스 엔지니어링 (2)

- 리버스 엔지니어링은 불법인가?
  - Wine, ReactOS에서 이미 폭넓게 활용
  - "상호운영성"을 위한 행위는 합법
  - 유럽, 미국에서는 합법, 국내는...?
- 걸림돌
  - IOCP와 같은 특허 기술을 우회할 방법 강구

점진적 이전

- 기본 NetBSD 시스템으로부터 시작
  - 크로스 툴체인 개발 환경 구축
- 커널 서브시스템/사용자 API 구현
  - 자체 빌드/명령행 프로그램 실행 가능
- XFree86 기반 윈도우 시스템 구현
- 고유 바이너리를 PE로 전환
  - Cygwin과 유사한 Unix/Windows 공존 환경
- 그래픽 인스톨러, 각종 액세서리 추가
- (최종) 완전 Windows 호환 운영체제

현재 상황

- KLDP.net에 프로젝트 입주
  - 두 개의 메일링 리스트
  - Win2k 힙 관리자 소스 코드 공개
  - MinGW-NetBSD 크로스 툴체인 공개
- 내부 개발 상황
  - 1000개에 가까운 NTDLL/KERNEL32/MSVCRT API 구현
  - 커널 기반 1:1 스레딩 모델 구현/시험중
  - 첫번째 완전 릴리즈는 11월경으로 예정

당면 과제

- 메일링 리스트 활성화
  - 메일링 리스트는 오픈 소스 문화의 중심
  - 게시판 중심의 국내 정서에 부합할 수 있는 방법은?
- API 검증 인력 확보
- 버전 관리 효율화
  - CVS의 고질적 문제
  - Subversion? arch?