신기술/기능 요구사항 지원 보고서
[제목: 리눅스 기반 개발환경]

한국소프트웨어진흥원

공개SW기술지원센터

<Revision 정보>

일자	VERSION	변경내역	작성자
2007.03.06	0.1	초기 작성	김상운

목 차

1. 문서 개요

본 문서는 일반적인 리눅스 기반의 개발환경에 대해서 관련 개발도구와 기타 환경에 대한 정보를 제공하고자 작성된 문서이다.

가. 문서의 목적

다음과 같은 세부적인 목적을 달성하기 위하여 작성되었다.

0 리눅스 기반의 개발 도구 설명

0 리눅스용 컴파일러 및 기타 적용 방안 소개

나. 본 문서의 사용방법

다음과 같은 방법으로 사용할 수 있다.

0 본 문서는 일반적인 리눅스 기반에서 사용할 수 있는 개발환경에 대해서 설명하고 있다.

0 개발기반이 되는 배포판에 따라 사용법이 조금씩은 다를 수 있으니 이점 유의하십시오.

2. 신기술/기능 요구사항

가. 대상 업체

구 분		비 고
기관명	자체 재작	
담당자		
연락처		
회사 위치		

나. 요구 사항

항목	지원 내용	비고
리눅스 기반 개발환경	● 소스코드 편집도구 소개 ● 컴파일러 소개 ● 기타 리눅스 기반 개발환경 소개	

3. 개발환경

가. 개요

리눅스 기반에서 리눅스 관련 개발을 위해서는 다양한 개발환경이 있다. 일반적으로 사용되는 개발환경을 및 도구들을 다음과 같이 나열해 보고 각 도구들에 대해서 간략하게 설명하고자 한다.

[1] GNU Toolkit
1) Binutils
2) GCC
3) GDB
4) c, c++ library

[2] 디버깅 도구
1) GDB
2) DDD
3) kdbg
4) ldd
5) strace
6) ltrace
7) checker

[3] 시간측정, 성능평가도구
1) time
2) gprof
3) calls

[4] 자동 컴파일링 도구
1) make
2) m4
3) Automake
4) Autoconf

[5] 인터페이스 만들기 도구
1) Xt
2) Motif/Lesstif
3) Xaw3D
4) Tcl/Tk
5) QT
6) GTK+

[6] 버전 관리 도구

1) CVS
2) RCS
3) SCCS

[7] 파일 패치 & 들여쓰기

1) patch
2) diff
3) indent

나. GNU toolkit

1) Binutils

여기에는 어셈블러(as), 링커(ld) 를 비롯하여 많은 바이너리 도구들이 포함되어 있다.
www.gnu.org 에서 내리는 간단한 요약을 적어 본다.
The GNU Binutils are a collection of binary tools. The main ones are:

- ld - the GNU linker.
- as - the GNU assembler.
- addr2line - Converts addresses into filenames and line numbers.
- ar - A utility for creating, modifying and extracting from archives.

(정적 라이브러리를 만드는 유틸리티이다. 다음과 같이 만든다.
ar rs libxxx.a a.o b.o c.o
s 옵션을 넣으면 ranlib 를 실행하지 않아도 된다.)

- c++filt - Filter to demangle encoded C++ symbols.
- gprof - Displays profiling information. (프로파일러)
- nlmconv - Converts object code into an NLM.
- nm - Lists symbols from object files. (오브젝트 파일의 심볼들을 나열해 준다.)
- objcopy - Copys and translates object files. (오브젝트 파일을 복사, 변환한다.)
- objdump - Displays information from object files. (오브젝트 파일을 덤프하여 여러가지 정보를 보여준다. 어셈블리어도 보여준다.)
- ranlib - Generates an index to the contents of an archive. (정정 라이브러리를 만들고 나서 인덱스 파일을 라이브러리 처음에 생생시켜 준다.)
- readelf - Displays information from any ELF format object file. (ELF 포맷의 오브젝트 파일의 정보를 디스플레이한다.)
- size - Lists the section sizes of an object or archive file. (#size any_program 이라고 명령하면 text , data , bss, dec 등의 섹션 크기를 출력해 준다.)
- strings - Lists printable strings from files. (파일로부터 출력 가능한 스트링을 나열해 준다.)
- strip - Discards symbols. (심볼들을 제거한다.)
- windres - A compiler for Windows resource files. (모른다.)

2) GCC(GNU C/C++ Compiler)

GNU의 C/C++ 컴파일러. 리눅스 기반에서 가장 대표적으로 사용되는 컴파일러

3) GDB(GNU DeBugging Tool)

커멘드라인 기반의 디버깅 도구

4) c, c++ library

리눅스의 표준 C언어 라이브러리는 'glibc'.

다. 디버깅 도구

1) gdb, ddd, kdbg

이것들은 gdb는 기본 디버거이고, ddd와 kdbg는 gdb를 활용한 GUI 프론트 엔드일 뿐이다. 하지만 GUI화 되어 다양한 기능들을 가지고 있을 것이다. 주로 많이 사용되는 GUI기반의 툴은 ddd가 가장 많이 쓰인다.
kdbg는 kde기반에서 GUI 디버깅 도구이며, kdevelop이라는 QT용 IDE툴의 기본 디버거로 적용되어 있다.

2) ldd - 공유 라이브러리 의존성을 출력해 준다.

3) strace / ltrace

strace는 사용하는 시스템 콜을 추적하여 주는 것이고, ltrace는 라이브러리 호출을 추적하여 주는 것이다.

4) checker

이것은 프로그램을 코딩하다가 메모리 할당 루틴에서 문제가 있는 것 같으면, 사용하는 것이다. 컴파일하기 전에 -lchecker 라는 옵션을 주면 문제가 있는 메모리할당 루틴을 checker 가 검사하여 원인을 보고하여준다.

라. 시간 측정, 성능 평가 도구

1) time

실행시키는 프로그램의 수행 시간을 측정해 준다. time은 시스템 유틸리티이지만, 같은 이름의 라이브러리 함수가 존재한다.

2) gprof

기본적으로 GNU binutil에 포함되어 있는 프로파일러이다. 즉, 코드 안에서 병목 현상을 일으키는 곳을 파악할 수 있다. 각 함수가 얼마나 자주 호출되는지, 각 함수에서 소요된 시간이 얼마나 되는지 등 실행한 프로그램의 자세한 목록을 보여주는 도구이다. 컴파일 할때 -pg 옵션을 주고 해야 한다. 컴파일 후 실행하라. 정상적으로 종료하면, 현재 디렉토리에 gmon.out 이라는 파일을 내어 놓는다. 이 파일안에 실행 프로파일 정보가 들어 있으며 gprof를 사용하여 통계값을 볼 수 있다.

1.

5) calls

c 소스 코드 안에서의 호출 관계를 계층 구조로 보여 준다. 호출된 모든 함수의 인덱스를 만들거나 프로그램 구조에 대한 계층 구조 보고서를 작성할 유용하다.

마. 자동 컴파일링 도구

1) make - makefile이라는 컴파일 설정 파일을 이용하다 용량이 큰 소스들에 대해서 통합적으로 컴파일 할 수 있도록 해 준다.

2) m4 / automake / autoconf - 유닉스기반의 시스템에서 소프트웨어를 배포할 때, 특히나 플랫폼에 독립적인 프로그램을 배포할 때 꼭 필요한 것들이다. make 파일을 작성하는 것이 또 지루한 작업이기 때문에 그것을 자동으로 해주는 툴이 automake이다. autoconf는 configure 라는 스크립트를 자동으로작성해 주는 그런 툴이다. 우리가 프로그램을 설치할때 리눅스에서, 가장 먼저, ./configre 하지 않는가? 그 configure 스트립트를 만들어 주는 것이 autoconf 라는 툴이다. 그러면 configure 라는 스트립트는 여러가지를 시스템에 물어보고, 알맞은 makefile을 만들어 주는 것이다. 그러면 우리는 #make 만 쳐서 컴파일을 시키는 것이다. m4는 매크로 프리프로세서라는 것이다. aotoconf에서 내부적으로 사용되는 것이다.

바. 인터페이스 만들기 도구

1) Xt - X library의 기본 툴킷.

2) Motif / Lesstif - 유닉스상에서 전통적으로 인기있는 윈도우 툴킷이었는데, 상용이라서 오픈소스의 다음세대(QT, GTK+)들에게 대세가 넘어갔다. 그리고 Lesstif라는 오픈소스 구현체가 나왔는데 대세는 이미 QT, GTK+로 넘어갔다.

3) Xaw3D - 표준 아데나(Athena)위젯의 변형 버전으로써 마치 모티프와 같은 스타일의 3D 효과를 준다.

4) Tcl/Tk - 창, 버튼, 스크롤바 등 기존의 프로그램에서 사용하고 있는 X 기반의 완전한 인터페이스를 만들 수 있다.

5) Qt - 노르웨이 회사인 트롤 테크(Troll Tech)가 만든 C++ GUI 툴킷이며 윈도우와 가장 유사한 환경을 제공하는 멀티플랫폼 툴킷이다. 현재 리눅스, 윈도우즈, 맥에서도 적용가능하다.

6) GTK+ - 원래 김프라는 이미지 처리 프로그램을 위해 만든 C GUI 툴킷이다. 소스코드가 QT에 비해 훨씬 심플, 간결하다고 하며 한글처리 또한 쉽게 할 수 있다.

사. 버전 관리 도구

1) CVS(Concurrent Versioning system) - 현재 리눅스에서 거의 표준으로 자리 잡고 있는 버전관리 도구이다. 인터넷으로 전세계 개발자들이 함께 개발할 수 있게끔 해주는 도구이다. 현재 거의 모든 오픈 소스 프로젝트들이 CVS로 개발되고 있다.

2) RCS(Revision control system) - CVS와 많은 것이 비슷하다. 하지만 RCS로는 인터넷에 연결된 그런 분산 개발보다는 한 그룹내에서 개발 할 때 사용한다.

3) SCCS(Source Code Control System) - 카네기 멜론 대학에서 개발된 버전관리 도구이다.

아. 파일 패치 & 들여 쓰기

1) patch , diff - 정기적으로 갱신되는 프로그램이 있고 프로그램이 매우 많은 소스 파일로 이루어져 있는 경우 diff 매번 전체 소스 배포 파일을 내놓는 것이 적절치 않을 때가 있다. 이 때 가장 좋은 방법이 각 버전마다 변경된 부분만을 patch(펄의 창시자인 래리 월이 만든)로 갱신하는 것이다. 즉, patch 는 한 버전에서 다음 버전으로 갱신하려고 상황에 맞게 파일을 변환시켜주는 프로그램이며 diff 와 함께 사용된다.

```
<사용법>
patch -pNUM <patchfile
diff [option] from-file to-file
```

가) 예제1. 소스가 단순 파일 하나일 때

hello1.c 가 있고, 다음 버전인 hello2.c 가 있다고 하자.
우선 hello.patch 라는 패치 파일을 만들어 내려면 다음과 같이 한다.

```
#diff -c hello1.c hello2.c > hello.patch
```

이렇게 만든 패치 파일을 배포하면 되는 것이고 이 패치 파일을 다운 받은 사용자는 다음과 같이 하여 hello1.c 소스에 패치를 가한다.

```
#patch < hello.patch
```

나) 예제2. 소스가 디렉토리 구조일 때

hello1 라는 디렉토리가 있고, 다음 버전인 hello2라는 디렉토리가 있다고 하자.
우선 hello.patch 라는 패치 파일을 만들기 위해서는 다음과 같이 한다.

```
#diff -cr hello1 hello2 > hello.patch
( 옵션 -r 은 recursive를 의미 )
```

이렇게 만든 패치 파일을 배포하면 되는 것이고, 이 패치 파일을 다운 받은 사용자는 다음과 같이 하여 hello1 디렉토리 구조에 패치를 가한다.

```
#patch -p0 < hello.patch
```

2) indent - 소스코드에서 각 문맥에서 들여쓰기하여 소스코드를 보기 쉽게 정렬해 준다.