Locales mini-HOWTO

Peter Joot, *joot@ecf.toronto.edu* <mailto:joot@ecf.toronto.edu> v1.3, 1996년 6월 6일

이 만 용 *geoman@nownuri.net* <mailto:geoman@nownuri.net>

이 문서는 리눅스에서 locale 를 쓰기 위해 설정하는 방법을 설명한다.

목 차

제 1 절 소개

여기서 나는 localdef 를 인스톨하기 위하여 무엇을 했으며 어떻게 locale을 컴파 일했는지 그리고 그 경험에 대해서 이야기식으로 진행하려고 한다. 나는 단지 재미로 이 일에 뛰어들었고 다른 사람들도 이 일에 흥미를 가질 것이 라고 생각했다. 일단 설정만 되면 여러분의 선택한 로케일을 가지고 NLS 지원의 어플리케이션을 사용할 수 있다. 조금 있으면 로케일 지원은 표준 배포판의 일부 가 될 것이고 이 미니 하우투 문서에도 많은 내용이 추가되리라 본다.

제 2 절 도대체 로케일(locale)이란 무엇인가?

로케일은 여러분이 여러분의 프로그램 속으로 하드코딩을 해서는 안되는 언어/문 화 관련의 사항을 다룬다. 만약 다양한 로케일을 설치해두었다면 다음 환경 변수 설정을 통하여 로케일을 지원하는 프로그램의 실행 방식을 선택할 수 있다.

기본 로케일은 C 또는 POSIX 이며 libc 에 하드코딩되어 있다.

LANG

이 세팅을 통해 로케일을 설정한다. 하지만 LC_xxxx 환경 변수에 의해 변화될 수 있다.

LC_COLLATE

정렬 순서

LC_CTYPE

문자 정의, 대문자, 소문자,... 이들은 toupper, tolower, islower, isdigit 등의 함수에 사용된다.

LC_MONETARY

금액을 표현하는 형식 정보를 포함하고 있다. 천단위를 구분하는 문자, 소수점 문자, 그리고 금액 표시 문자, 그 위치 등의 정보를 제시한다.

LC_NUMERIC

숫자를 표현함에 있어, 천단위, 소수점, 숫자 그룹핑을 다룬다.

LC_TIME

시간과 날짜의 표현형식을 지정한다. 주와 달에 대한 명칭과 약어를 다룬 다.

LC_MESSAGES

예/아니오 표현

LC_ALL

로케일을 지정한다. 하지만 다른 LC_xxxx 으로 변화될 수 있다. 여기 몇 가지 로케일에 대한 예가 있다.

en_CA

영어권 카나다

en_US

미국 영어권

de_DE

독일의 독일어

fr_FR

프랑스의 프랑스어

여러분이 프로그램을 짜는데 있어 그것이 국제적으로 사용되기 위해서는 로케일 을 이용해야 한다. 가장 중요한 이유 중 하나는 모든 사람이 여러분과 같은 문자 를 사용하지는 않는다는 사실이다. 여러분의 프로그램에서 다음과 같은 식으로 사용하지 말라.

/* check for alphabetic characters */ if (((c >= 'a') && (c <= 'z')) || ((c >= 'A') && (c <= 'Z'))) { ... }

여러분이 이렇게 모든 화일을 ASCII 화일이라고만 가정하고 만든다면 사용자의 로케일에 따른 코드 페이지 정의를 존중하지 않게 되는 것이다.

예를 들어 독일어 환경에서 쓰이는 움라우프 문자 등이 제외된다고 말할 수 있다. 여러분은 대신 isalpha() 과 같은 로케일 인식 함수를 사용해야 한다. 여러분이 US-ASCII 알파벳 문자만을 요구한다 할 지라도 여전히 isalpha() 함수를 사용하기 바란다. 하지만 우선 setlocalte(LC_TYPE,"C")라고 하든지 아니면 LANG, LC_TYPE, LC_ALL 등의 환경 변수를 "C"로 설정하면 된다. 로케일은 상당한 유연성을 제공하며 프로그래머들이 ASCII 기반의 C 프로그램을 잘못 만들어왔다고 생각한다.

예를 들어 여러분은 문자의 코드 위치를 가정할 수 없다. 여러분은 A 라는 문자의 코드 위치를 0x41 이 아닌 0xC1 으로 정의하는 문자표 화일을 만들 수도 있다. 이것은 메인프레임에서 사용된는 IBM 코드 페이지 37에서의 A 가 갖는 위치이다. 그리고 전자는 바로 US-ASCII, iso8859-x 등에서 사용되고 있는 방식이다. 기본적인 아이디어는 서로 다른 사람들이 다른 언어를 사용하며 정렬 순서도 다르 고 다른 코드 페이지를 사용하며 다른 나라에 살고 있다는 사실로부터 나온다. 바로 로케일과 로케인 인식 프로그램이 그러한 문제에 대한 해결책을 제시하고 있 다. 많은 작업이 필요한 것은 아니고 단지 프로그램 작성시 약간 다른 사고방식을 요구한다.

제 3 절 주의 사항

[] 내 머신에서 로케일을 설정하기 위하여 약간의 업그레이드가 필요했다. *ftp.tu-clausthal.de:/pub/linux/SLT/nls* `<ftp://ftp.tu-clausthal.de:/pub/linux/SLT/nls>` 에는 a.out 버전의 로케일과 local- def(nlsutils-0.5.tar.gz 화일 안에)이 들어있다. ELF 시스템이 아니거나 ELF 를 사용하지 않으려는 사람에게 유용하다. 다른 곳에 아마도 nls 유틸 패키지가 있을 것이다. 하지만 찾지 못했다. 독립적인 로케일, localdef가 있는지 모르겠으며 그러한 libc 를 가지고 있는지도 모르겠다. 이 하우투 문서는 본인이 libc 등을 업그레이드한 개인적인 경험일 분이다. 여러분이 이 작업을 하려면 본인처럼 ELF 시스템을 운영하거나 ELF 시스템으 로 업그레이드해야 할 것이다.

- 본인이 한 시스템 업그레이드는 a.out 에서 ELF 업그레이드하기 위해 필요한 것들이었다. ELF 시스템이 아니라면 여러분은 일단 최근 ELF HOWTO 문서를 구 해야 할 것이다. 상당히 좋은 문서이며 libc, ld.so, 그리고 다른 ELF 시스템 인스톨, 업그레이드에 대한 많은 지식을 전해줄 것이다.
- 여러분이 인스톨하는 모든 것에 대해서 릴리즈 노트나 README 를 읽어보라. 여기서 본인이 말한 바를 제대로 이해하지 못하거나 그로 인해 시스템을 망쳤 다고 본인을 원망하지는 말기 바란다.
- 새로운 libc, ld.so 를 잘못 설치하는 경우 부팅 자체가 안될 수 있다. 일단은 부트 디스크를 하나 가지고 중요한 자료는 백업을 해두는 것이 좋다.

제 4 절 여러분에게 필요한 사항

몇 개의 자료를 다운로드받을 필요가 있다. locale 소스 화일을 제외한 대부분은 sunsite.unc.edu, tsx-11.mit.edu, 또는 로컬 미러 사이트에서 구할 수 있다.

```
[] 로케일, 문자표 소스 - localdef 를 가지고 컴파일하게 될 것이다.
[] libc-5.2.18.bin.tar.gz --- ELF C 공유 라이브러리와 수학 라이브러리
[] libc-5.2.18.tar.gz --- ELF 라이브러리에 대한 소스 코드. localdef 를 컴파
    일하기 위해서는 바로 이 화일이 필요하다.
[] make-3.74.tar.gz -- dirent 버그를 고친 버전이 필요하다.
[] release.libc-5.2.18 --- make 에 대한 패치를 포함한 릴리즈 노트
[] ld.so-1.7.12+ -- 동적 링커
[] ELF gcc-2.7.2+ --- 컴파일에 사용
[] ELF 커널 1.1.92+ 또는 ELF 커널 1.3.40+ -- 컴파일에 사용
[] binutils 2.6.0.2+ --- 컴파일에 사용
```

로케일 소스를 얻을 수 있는 곳은 많다. 공개 로케일과 문자표를 다음과 같은 사이트에서 얻을 수 있었다. dkuug.dk:/i18n/WG15-collection/locales <*ftp.dkuug.dk/i18n/WG15-collection/locales* `<ftp://ftp.dkuug.dk/i18n/WG15-collection/locales>`> dkuug.dk:/i18n/WG15-collection/charmaps < *ftp.dkuug.dk/i18n/WG15-collection/charmaps* `<ftp://ftp.dkuug.dk/i18n/WG15-collection/charmaps>`>

제 5 절 모든 것을 인스톨하기

이것 모두가 본인이 인스톨한 것이다. 본인은 이미 ELF 시스템(컴파일러, 커널) 을 가지고 있었다.

1. 우선은 tar xvzf binutils-2.6.0.2.bin.tar.gz -C / 하여 바이너리 유틸리티를 설치하였다.
2. 다음으로는 동적 링커를 설치하였다.

   tar zxf ld.so-1.7.12.tar.gz -C /usr/src cd /usr/src/ld.so-1.7.12 sh instldso.sh
3. . 그 다음 나는 lib 를 설치하였다. 자세한 설명은 release.libc-5.2.18 을 보기 바란다.

   rm -f /usr/lib/libc.so /usr/lib/libm.so rm -f /usr/include/iolibio.h /usr/include/iostdio.h rm -f /usr/include/ld_so_config.h /usr/include/localeinfo.h rm -rf /usr/include/netinet /usr/include/net /usr/include/pthread tar -xzf libc-5.2.18.bin.tar.gz -C /
4. 새로운 공유 라이브러리의 위치를 알리기 위하여 ldconfig -v 를 실행
5. make 과 다른 프로그램에 대한 버그 패치가 libc 에 들어있었다. 여기서 나는 make 를 다시 컴파일하여 인스톨하였다.

   tar zxf make-3.74.tar.gz -C /usr/src cd /usr/src/make-3.74 patch < /whereever_you_put_it/release.libc-5.2.18 configure –prefix=/usr sh build.sh ../make install cd .. rm -rf make-2.74
6. 여기서 localdef 를 컴파일하여 설치하였다.

   mkdir /usr/src/libc tar zxf libc-5.2.18.tar.gz -C /usr/src/libc cd /usr/src/libc cd include ln -s /usr/src/linux/include/asm . ln -s /usr/src/linux/include/linux . cd ../libc ./configure

   # 이 두 가지 make 과정이 필요한지는 모르겠으나 안전하게 해두도록 하자.

   make clean ; make depend cd locale make programs mv localedef /usr/local/bin mv locale /usr/local/bin
7. localdef 가 찾을 수 있는 곳에 문자표를 설치한다. dkuug.dk FTP 사이트로 부터 charmaps.tar, locales.tar 라는 화일로 구한 것을 사용하였다.

tar xf charmaps.tar -C /tmp mkdir /usr/share/nls mkdir /usr/share/nls/charmap mkdir /usr/share/locale mv /tmp/charmaps/* /usr/share/nls/charmap rm -rf /tmp/charmaps tar xf locales.tar -C /usr/share # put these where you want.

```
# 어떤 로케일 소스는 'copy'를 사용하며 따라서 다른 로케일에 대한
  # 의존성을 가진다. 본인은 grep, cut, sed 를 사용하여 makefile 을
  # 이렇게 해주면 /usr/share/locale 디렉토리에 모든 로케일 객체를
  # 만들어 줄 것이다.
  #
  localedef -ci locales/en_DK -f ISO_8859-1:1987 en_DK
  localedef -ci locales/sv_SE -f ISO_8859-1:1987 sv_SE
  localedef -ci locales/fi_FI -f ISO_8859-1:1987 fi_FI
  localedef -ci locales/sv_FI -f ISO_8859-1:1987 sv_FI
  localedef -ci locales/ro_RO -f ISO_8859-1:1987 ro_RO
  localedef -ci locales/pt_PT -f ISO_8859-1:1987 pt_PT
  localedef -ci locales/no_NO -f ISO_8859-1:1987 no_NO
  localedef -ci locales/nl_NL -f ISO_8859-1:1987 nl_NL
  localedef -ci locales/fr_BE -f ISO_8859-1:1987 fr_BE
  localedef -ci locales/nl_BE -f ISO_8859-1:1987 nl_BE
  localedef -ci locales/da_DK -f ISO_8859-1:1987 da_DK
```

```
localedef -ci locales/kl_GL -f ISO_8859-1:1987 kl_GL
localedef -ci locales/it_IT -f ISO_8859-1:1987 it_IT
localedef -ci locales/is_IS -f ISO_8859-1:1987 is_IS
localedef -ci locales/fr_LU -f ISO_8859-1:1987 fr_LU
localedef -ci locales/fr_FR -f ISO_8859-1:1987 fr_FR
localedef -ci locales/de_DE -f ISO_8859-1:1987 de_DE
localedef -ci locales/de_CH -f ISO_8859-1:1987 de_CH
localedef -ci locales/fr_CH -f ISO_8859-1:1987 fr_CH
localedef -ci locales/en_CA -f ISO_8859-1:1987 en_CA
localedef -ci locales/fr_CA -f ISO_8859-1:1987 fr_CA
localedef -ci locales/fo_FO -f ISO_8859-1:1987 fo_FO
localedef -ci locales/et_EE -f ISO_8859-1:1987 et_EE
localedef -ci locales/es_ES -f ISO_8859-1:1987 es_ES
localedef -ci locales/en_US -f ISO_8859-1:1987 en_US
localedef -ci locales/en_GB -f ISO_8859-1:1987 en_GB
localedef -ci locales/en_IE -f ISO_8859-1:1987 en_IE
localedef -ci locales/de_LU -f ISO_8859-1:1987 de_LU
localedef -ci locales/de_BE -f ISO_8859-1:1987 de_BE
localedef -ci locales/de_AT -f ISO_8859-1:1987 de_AT
localedef -ci locales/sl_SI -f ISO_8859-2:1987 sl_SI
localedef -ci locales/ru_RU -f ISO_8859-5:1988 ru_RU
localedef -ci locales/pl_PL -f ISO_8859-2:1987 pl_PL
localedef -ci locales/lv_LV -f BALTIC lv_LV
localedef -ci locales/lt_LT -f BALTIC lt_LT
localedef -ci locales/iw_IL -f ISO_8859-8:1988 iw_IL
localedef -ci locales/hu_HU -f ISO_8859-2:1987 hu_HU
localedef -ci locales/hr_HR -f ISO_8859-4:1988 hr_HR
localedef -ci locales/gr_GR -f ISO_8859-7:1987 gr_GR
```

제 6 절 이제 무엇을 할 것인가?

이렇게 설치하고 난 후 여러분은 로케일을 사용할 수 있게 된다. 여기 아주 간단한 예가 있다.

```
/* test.c : 로케일이 로딩되고 사용되는지 테스트 */
  #include <locale.h>
  #include <stdio.h>
  #include <time.h>
main(){
          time_t t;
          struct tm * _t;
          char buf[256];
        time(&t);
          _t = gmtime(&t);
        setlocale(LC_TIME,"");
          strftime(buf,256,"%c",_t);
        printf("%s\n",buf);
  }
```

로케일 프로그램을 사용하여 하여 현재 로케일 환경변수가 어떤지 알아볼 수 있다.

```
$ # 서로 다른 몇 가지 로케일 설정을 하여 위 프로그램을 수행시킨다.
$ gcc -s -o Test test.c
$ # 로케일이 무엇인지 알아본다.
$ locale
LANG=POSIX
LC_COLLATE="POSIX"
LC_CTYPE="POSIX"
LC_MONETARY="POSIX"
LC_NUMERIC="POSIX"
LC_TIME="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
$ # 지루하기 짝이 없는 C 로케일을 사용하고 있다. -_-
$ # 영어권 캐나다로 바꾸어보자.
$ export LC_TIME=en_CA
$ Test
Sat 23 Mar 1996 07:51:49 PM
$ # 이번에는 프랑스어권 캐다나로 바꾸어보자.
$ export LC_TIME=fr_CA
$ Test
sam 23 mar 1996 19:55:27
```

제 7 절 catopen 버그 고치기

로케일을 인스톨하게 되면 리눅스 libc 에 있는 catopen 명령에 있는 버그(기능?) 을 고치게 된다. 메세지 목록을 사용하는 프로그램을 만든 후 독일어 목록을

/home/peeter/catalogs/de_DE 에 만들어 넣으라.

de_DE 로케일을 인스톨하기 전에 다음을 수행한다.

export LC_MESSAGES=de_DE export NLSPATH=/home/peeter/catalogs/%L/%N.cat:$NLSPATH 독일어 메세지는 열리지 않고 catget 의 기본 메세지가 사용된다.

이 이유는 catopen 이 올바른 메세지 범주를 얻어내기 위해 setlocale 를 호출하 하는데 setlocale 은 환경변수가 제대로 세팅되어 있음에도 불구하고 실패하 기 때문이다.

catopen 은 NLSPATH의 모든 "%L"에 대하여 C 로 교체하려고 한다. 로케일을 설치하지 않아도 여러분의 메세지 목록을 사용할 수는 있다. 하지만 NLSPATH를 정확히 다음과 같이 명시해주어야 할 것이다.

export NLSPATH=/home/peeter/catalogs/de_DE/%N.cat:$NLSPATH 하지만 이렇게 되면 로케일 범주에 대한 환경변수를 사용할 필요성 자체가 사라 지는 것이다.

제 8 절 질문과 답

이 섹션은 지금으로서는 FAQ 라고 할 수 없겠지만 앞으로 FAQ 로 자라날 것이다.

8.1 msgcat 질문

나는 리눅스 사용자이며 다음과 같은 시험 프로그램을 만들었다.

```
    #include <stdio.h>
    #include <locale.h>
    #include <features.h>
    #include <nl_types.h>
  main(int argc, char ** argv)
    {
            nl_catd catd;
          setlocale(LC_MESSAGES, "");
            catd = catopen("msg", MCLoadBySet);
            fprintf(stderr,catgets(catd, 1, 1, "locale message fail\n"));
            catclose(catd);
    }
```

$ msg.m $set 1 1 locale message pass\n

If I use absolute path in catopen like

내가 만약 catopen("/etc/locale/msg.cat",MCLoadBySet); 와 같이 절대 경로를 사용하면 옳은 결과를 얻어내지만 위처럼 사용하면 결과값 -1 즉 실패하게 된다.

8.2 msgcat 답변

이 질문은 지난 섹션에서 답변이 된 문제이긴 하지만 몇 가지 부가적인 정보를 말할 기회라고 생각한다.

여러분의 메세지 목록을 넣을 수 있는 곳은 아주 많다. 환경변수에서 NLSPATH를 명시적으로 설정하지 않았다 하더라도 다음과 같이 libc 에 정의되어 있다.

```
  $ strings /lib/libc.so.5.2.18 | grep locale | grep %L
    /etc/locale/%L/%N.cat:/usr/lib/locale/%L/%N.cat:/usr
    /lib/locale/%N/%L:/usr/share/locale/%L/%N.cat:/usr/
    local/share/locale/%L/%N.cat

따라서 다음 중 하나를 세팅해주면 :
  $ export LC_MESSAGES=en_CA
    $ export LC_ALL=en_CA
    $ export LANG=en_CA

위에서 보여진 NLSPATH와 환경변수에 의거하여 catopen("msg", MCLoadBySet); 는
  다음 디렉토리 중 하나에 메세지 목록이 있는 경우 작동하게 된다.
  /etc/locale/en_CA/msg.cat
    /usr/lib/locale/en_CA/msg.cat
    /usr/lib/locale/msg/en_CA
    /usr/share/locale/en_CA/msg.cat
    /usr/local/share/locale/en_CA/msg.cat
```

This, however, will not work if you don't have the en_CA locale

하지만 여러분이 en_CA 로케일을 설치하지 않았다면 제대로 동작하지 않는다. 왜냐하면 setlocale 가 실패하여 catopen 루틴의 %L을 en_CA 가 아닌 C 로 교체 하기 때문이다.

제 9 절 끝내는 말

자, 이것이 전부다. 이 문서가 여러분에게 약간이라고 도움이 되었기를 바란다. 로케일 인식 프로그램을 짜는데 대한 다른 정보를 찾을 수 있는 곳이 많이 있을 것이다. libc에 대한 info 페이지에도 약간의 정보가 있고 맨페이지에도 있다. 약간 웹을 돌아다니다 보면 많은 정보를 얻을 수 있으리라 본다. 국제화된 프로그램에 대한 좋은 정보 원천을 찾게 되고 본인도 알기 바란다면, 여기다 적어놓기로 하겠다.