การใช้ภาษาไทยบนลินุกซ์

เทพพิทักษ์ การุญบุญญานันท์

สารบัญ

1 ข้อกำหนดธรรมเนียมท้องถิ่น (โลแคล) 5

1.1 โลแคลคืออะไร 5

1.1.1 POSIX Locale 5

1.1.2 ISO/IEC 14652 Cultural Convention 6

1.2 ตัวแปรระบบสำหรับการตั้งค่าโลแคล 6

1.3 การตั้งโลแคลอัตโนมัติ 7

1.3.1 โลแคลทั้งระบบ 7

1.3.2 โลแคลเฉพาะผู้ใช้ 7

1.3.3 โลแคลขณะล็อกอิน 7

2 การตั้งค่าแป้นพิมพ์และการป้อนข้อความ 9

2.1 ระบบป้อนอินพุตบนเอกซ์วินโดว์ 9

2.2 การตั้งค่า XKB 9

2.3 การตั้งค่า XIM 11

2.4 ระบบป้อนอินพุตของ GTK+ 2 11

3 การติดตั้งฟอนต์ 13

3.1 ระบบฟอนต์บนเอกซ์วินโดว์ 13

3.1.1 ระบบรายชื่อฟอนต์ของเอกซ์เซิร์ฟเวอร์ 13

3.1.2 การใช้ฟอนต์ผ่านฟอนต์เซิร์ฟเวอร์ 15

3.2 การติดตั้งฟอนต์บนเอกซ์วินโดว์ 15

3.2.1 การติดตั้งฟอนต์บิตแมพ 16

3.2.2 การติดตั้งฟอนต์ Type 1 17

3.2.3 การติดตั้งฟอนต์ TrueType 17
3.3 ฟอนต์ที่ฝั่งไคลเอนต์ 18

4 การพิมพ์เอกสารภาษาไทย **21**
4.1 ระบบการพิมพ์บนลินุกซ์ 21
4.2 Ghostscript 22
4.3 Mozilla กับ Xprint 23
4.3.1 การตั้งค่า Xprint server 23
4.3.2 การ start Xprint server 24

5 การใช้ xiterm+thai **27**

6 การใช้ภาษาไทยใน LaTeX **29**
6.1 การใช้ภาษาไทยด้วย babel package 29
6.2 การตัดคำด้วย swath และ cttex 30
6.3 การคอมไพล์เอกสาร LaTeX 31

7 การใช้ภาษาไทยใน LyX **33**
7.1 การตั้งค่าฟอนต์สำหรับแสดงผล 33
7.2 การตั้งค่าแป้นพิมพ์ 33
7.3 การสร้างเอกสารภาษาไทยด้วย LyX 34

8 การใช้ภาษาไทยใน text console **35**
8.1 การติดตั้งฟอนต์ 35
8.2 การตั้งค่าแป้นพิมพ์ 36
8.3 การตั้งค่าเริ่มต้นอัตโนมัติ 37
8.4 แหล่งดาวน์โหลด 37

9 การแปลงรหัสข้อมูลภาษาไทย **39**
9.1 การแปลงรหัสข้อมูลด้วย iconv 39

บทที่

1

ข้อกำหนดธรรมเนียมท้องถิ่น (โลแคล)

1.1 โลแคลคืออะไร

ในยุคสมัยหนึ่ง โปรแกรมต่างๆ ที่เขียนขึ้นล้วนแต่ใช้ภาษาอังกฤษเป็นหลัก หรืออย่างดีที่สุดก็สนับสนุนภาษาในยุโรป ถ้าผู้ใช้ภาษาอื่นนอกจากนั้นต้องการใช้ภาษาของตนเอง ก็ต้องพัฒนาโปรแกรมขึ้นมาเอง หรือมิฉะนั้น ผู้ผลิตก็ต้องมาแก้โปรแกรมให้เป็น Thai Edition, Chinese Edition ฯลฯ และจะใช้ข้ามภาษาอีกไม่ได้เช่นกัน ซ้ำร้าย เมื่อออกซอฟต์แวร์รุ่นใหม่ออกมา ก็ต้องมาแก้โปรแกรมกันใหม่อยู่ร่ำไป

จนมาถึงยุคหนึ่ง พร้อมกับที่โลกเกิดกระแสโลกาภิวัตน์ (globalization) วงการซอฟต์แวร์ก็เกิดคำว่า *internationalization* ขึ้นมา เพราะความสามารถของซอฟต์แวร์ที่จะใช้งานได้ทั่วโลกมีความสำคัญมากขึ้น จนต้องจัดระบบในการสนับสนุนภาษาท้องถิ่นต่างๆ ให้มากที่สุดโดยไม่เปลืองแรงงานโดยไม่จำเป็น แนวคิดก็คือ แยกเอาส่วนของโปรแกรมที่ต้องแปรเปลี่ยนไปตามธรรมเนียมท้องถิ่นออกมาเป็นนามธรรม (abstract) โดยนามธรรมนั้นจะกลายเป็นรูปธรรมเมื่อเชื่อมต่อเข้ากับส่วนข้อกำหนดธรรมเนียมท้องถิ่นที่ทำงานจริง ซึ่งจะแตกต่างกันไปตามภาษาและท้องที่ ดังนั้น โปรแกรมที่ผ่านกระบวนการนี้จึงกลายเป็นโปรแกรมที่ “internationalized” คือสามารถทำงานได้ทุกท้องที่ โดยยังปรับเปลี่ยนพฤติกรรมไปตามท้องที่ได้โดยไม่ต้องแก้ไขเพิ่มเติม

งาน internationalization หรือที่เรียกย่อๆ ว่า i18n (ตัว i ตามด้วยอักษรอีก 18 ตัว ปิดท้ายด้วยตัว n) ถือเป็นงานในส่วนของการสร้างข้อกำหนดในโปรแกรม งานส่วนที่เหลือซึ่งขึ้นอยู่กับแต่ละท้องที่ก็คือการสร้างองค์ประกอบที่เรียกว่า *โลแคล (locale)* ซึ่งบรรยายหรืออิมพลีเมนต์ส่วนของโปรแกรมที่เป็นเอกลักษณ์ของท้องถิ่น กระบวนการสร้างโลแคลนี้ ก็เรียกกันว่า *localization (L10N)*

1.1.1 POSIX Locale

POSIX (Portable Operating System Interface) เป็นข้อกำหนดมาตรฐานของระบบปฏิบัติการ (โดยอิงยูนิกซ์) เพื่อให้โปรแกรมต่างๆ ที่สนับสนุนข้อกำหนดนี้สามารถทำงานได้ในทุกๆ ระบบปฏิบัติการที่เป็น POSIX

POSIX มีข้อกำหนดหลายส่วน ตั้งแต่ system call, standard library ไปจนถึงคำสั่งพื้นฐานต่างๆ เฉพาะในส่วนของ i18n นี้ POSIX ได้กำหนดโลแคลไว้ 6 หัวข้อ ได้แก่

1. LC_CTYPE: การจัดหมวดหมู่ของอักขระ
2. LC_COLLATE: การเรียงลำดับข้อความ
3. LC_TIME: รูปแบบปฏิทินและเวลา
4. LC_NUMERIC: รูปแบบการแสดงตัวเลข
5. LC_MONETARY: หน่วยเงินตราและรูปแบบการแสดงจำนวนเงิน
6. LC_MESSAGES: การแสดงข้อความทั่วไป

โดยชื่อของโลแคลที่ใช้ จะมีรูปแบบเป็น

lang[_*TERRITORY*[.*CHARSET*]]

โดยที่

lang	หมายถึงภาษาที่ใช้ (กำหนดรายชื่อไว้ใน ISO 639)
TERRITORY	หมายถึงประเทศที่ตั้ง (กำหนดรายชื่อไว้ใน ISO 3166-1)
CHARSET	หมายถึงรหัสอักขระที่ใช้

ตัวอย่างเช่น

en	หมายถึงภาษาอังกฤษ
en_US	หมายถึงภาษาอังกฤษในอเมริกา
en_US.ISO-8859-1	หมายถึงภาษาอังกฤษในอเมริกา ใช้รหัส Latin 1
th	หมายถึงภาษาไทย
th_TH	หมายถึงภาษาไทยในประเทศไทย
th_TH.TIS-620	หมายถึงภาษาไทยในประเทศไทย ใช้รหัส มอก. 620
th_TH.UTF-8	หมายถึงภาษาไทยในประเทศไทย ใช้รหัส UTF-8

1.1.2 ISO/IEC 14652 Cultural Convention

เมื่อเข้าสู่การกำหนดมาตรฐาน ISO สำหรับโลแคล หัวข้อทั้ง 6 ของ POSIX ได้ถูกนำมาใช้เป็นพื้นฐานทั้งหมด โดยแต่ละหัวข้อได้มีส่วนขยายเพิ่มเติมตามสมควร รวมทั้งได้มีหัวข้อโลแคลเพิ่มขึ้นอีก 6 หัวข้อ ได้แก่

1. LC_PAPER: ขนาดกระดาษมาตรฐาน
2. LC_MEASUREMENT: หน่วยวัดมาตรฐาน
3. LC_NAME: รูปแบบการเรียกชื่อ
4. LC_ADDRESS: รูปแบบที่อยู่ การจ่าหน้าซองจดหมาย
5. LC_TELEPHONE: รูปแบบหมายเลขโทรศัพท์
6. LC_VERSIONS: รุ่นของโลแคล

1.2 ตัวแปรระบบสำหรับการตั้งค่าโลแคล

การกำหนดโลแคลตาม POSIX นั้น จะยึดตามค่าตัวแปร environment ของโปรเซส ซึ่งตั้งค่าได้จากเชลล์ โดยลำดับความสำคัญเป็นดังนี้:

1. หาก LC_ALL มีค่า ให้ถือค่าตาม LC_ALL ทุกหัวข้อ
2. หาก LC_ALL ไม่ได้กำหนดค่าไว้ LC_CTYPE, LC_COLLATE, LC_TIME, LC_NUMERIC, LC_MONETARY, LC_MESSAGES ให้มีผลต่อโลแคลหัวข้อต่างๆ ตามที่ได้กำหนดค่าไว้
3. สำหรับหัวข้อที่ยังไม่กำหนดค่า หาก LANG มีค่า ให้ถือค่าตาม LANG
4. หาก LANG ก็ไม่ได้กำหนดค่าไว้ ให้ถือว่าค่าปริยายคือ C หรือ POSIX คือประพฤติตามพฤติกรรมปกติของภาษา C ตามข้อกำหนด POSIX

1.3 การตั้งโลแคลอัตโนมัติ

1.3.1 โลแคลทั้งระบบ

ผู้ดูแลระบบสามารถตั้งค่าโลแคลปริยายสำหรับระบบทั้งระบบได้ ซึ่งวิธีตั้งค่าจะขึ้นอยู่กับลินุกซ์ที่ใช้ เช่น

- **Red Hat และลินุกซ์ทะเล** ตั้งค่าที่ไฟล์ `/etc/sysconfig/i18n`
- **Debian** ใช้คำสั่ง `dpkg-reconfigure locales` หรือตั้งค่าที่ไฟล์ `/etc/environment`

1.3.2 โลแคลเฉพาะผู้ใช้

ผู้ใช้สามารถตั้งค่าแทนที่ค่าปริยายที่ผู้ดูแลระบบกำหนดให้เพื่อใช้เฉพาะบุคคลได้เช่นกัน และวิธีตั้งค่าที่ใช้ได้ทุกระบบก็คือ แก้ที่ไฟล์เริ่มแรกของเชลล์ที่ใช้ กล่าวคือ

- **bash** แก้ที่ `~/.bash_profile` สำหรับเชลล์ล็อกอิน หรือ `~/.bashrc` สำหรับเชลล์ทั่วไป
- **csh** แก้ที่ `~/.login` สำหรับเชลล์ล็อกอิน หรือ `~/.cshrc` สำหรับเชลล์ทั่วไป
- **tcsh** แก้ที่ `~/.login` สำหรับเชลล์ล็อกอิน หรือ `~/.tcshrc` หรือ `~/.cshrc` สำหรับเชลล์ทั่วไป

ส่วน Red Hat และลินุกซ์ทะเล ผู้ใช้สามารถกำหนดโลแคลเพื่อใช้เฉพาะบุคคลได้ที่ `~/.i18n` อีกด้วย

1.3.3 โลแคลขณะล็อกอิน

การล็อกอินผ่าน gdm หรือ kdm สามารถเลือกโลแคลขณะล็อกอินได้ในเมนู Language หรือภาษา

บทที่

2

การตั้งค่าแป้นพิมพ์และการป้อนข้อความ

2.1 ระบบป้อนอินพุตบนเอกซ์วินโดว์

การป้อนอินพุตผ่านแป้นพิมพ์บนเอกซ์วินโดว์ มี 2 ขั้นตอนหลัก คือ

1. แปลง scan code ของปุ่มที่กดให้เป็น keyboard symbol
2. แปลง keyboard symbol ให้เป็นรหัสอักขระ

เมื่อเรากดปุ่มบนแป้นพิมพ์ โปรแกรมที่เป็นเอกซ์ไคลเอนต์จะได้รับเหตุการณ์ (event) จากแป้นพิมพ์ โดยส่งมาเป็น scan code บอกตำแหน่งบนแป้นของปุ่มที่กด จากนั้น เหตุการณ์เหล่านี้จะถูกส่งให้ *X Input Method (XIM)* เพื่อตีความเหตุการณ์ต่อ โดยเริ่มจากองค์ประกอบที่เรียกว่า *X Keyboard (XKB)* จะถูกเรียกเพื่อแปลง scan code ของปุ่มกดให้เป็น keyboard symbol ซึ่งการแปลงดังกล่าวจะขึ้นอยู่กับผังของแป้นพิมพ์ เช่น เป็นแบบ qwerty, dvorak, เกษมณี หรือปัตตโชติ ฯลฯ

จากนั้น ลำดับของ keyboard symbol จึงค่อยถูกแปลให้เป็นรหัสอักขระตามโลแคลที่ใช้ และส่งเป็นข้อความอินพุตกลับไปให้โปรแกรมที่เป็นเอกซ์ไคลเอนต์อีกทอดหนึ่ง

ดังนั้น การป้อนภาษาไทยในเอกซ์วินโดว์จึงมีสองส่วนที่เกี่ยวข้องที่เราจะต้องปรับแต่งให้เหมาะสม ได้แก่ XKB และ XIM

2.2 การตั้งค่า XKB

ใน XFree86 จะมีผังแป้นพิมพ์เกษมณีสำหรับภาษาไทยมาให้อยู่แล้ว (ในรุ่นถัดๆ ไปจะมีเพิ่มอีกสองแบบ คือ มอก.820-2538 และปัตตโชติ) โดยจะเก็บไว้ที่ไฟล์ `/usr/X11R6/lib/X11/xkb/symbols/pc/th` และ "`th`" ก็เป็นชื่อของผัง XKB สำหรับภาษาไทย XFree86 ตั้งแต่รุ่น 4.3.0 เป็นต้นไปอนุญาตให้ผสมภาษาแป้นพิมพ์ได้ถึง 4 ภาษาพร้อมๆ กัน เราสามารถสั่งเลือกผัง XKB อังกฤษ-ไทยได้ด้วยคำสั่ง `setxkbmap` ดังนี้

```
setxkbmap us,th -option grp:alt_shift_toggle,grp_led:scroll
```

หรือหากจะตั้งให้เป็นค่าปริยาย ก็ตั้งได้ที่ไฟล์ `/etc/X11/XF86Config-4` โดยแก้ที่ `Section "InputDevice"` ในรายการที่ `Driver` เป็น `"Keyboard"` โดยแก้ตัวเลือก `"XkbLayout"` ดังนี้

```
Option "XkbLayout"  "us,th"
Option "XkbOptions" "grp:alt_shift_toggle,grp_led:scroll"
```

ตัวเลือก “`grp:alt_shift_toggle,grp_led:scroll`” เป็นการเลือกใช้ปุ่ม Alt-Shift เป็นปุ่มสลับภาษา และใช้ ScrollLock LED แสดงสถานะภาษา (ไฟจะติดเมื่อเป็นกลุ่มที่สองขึ้นไป ซึ่งในที่นี้ก็คือภาษาไทย)

ตารางที่ 2.1 และ 2.2 แสดงตัวเลือกที่เป็นไปได้ทั้งหมดสำหรับปุ่มสลับภาษาและการแสดงสถานะด้วย LED ของแป้นพิมพ์ตามลำดับ

ตารางที่ 2.1: ตัวเลือก XKB สำหรับปุ่มสลับภาษา

ตัวเลือก	ความหมาย
`grp:switch`	Alt ขวา เปลี่ยนภาษาเมื่อกดค้าง
`grp:lwin_switch`	Win ซ้าย เปลี่ยนภาษาเมื่อกดค้าง
`grp:rwin_switch`	Win ขวา เปลี่ยนภาษาเมื่อกดค้าง
`grp:win_switch`	Win เปลี่ยนภาษาได้ทั้งสองปุ่มเมื่อกดค้าง
`grp:toggle`	Alt ขวา สลับภาษา
`grp:lalt_toggle`	Alt ซ้าย สลับภาษา
`grp:caps_toggle`	CapsLock สลับภาษา
`grp:shift_toggle`	Shift คู่ สลับภาษา
`grp:alts_toggle`	Alt คู่ สลับภาษา
`grp:ctrls_toggle`	Ctrl คู่ สลับภาษา
`grp:ctrl_shift_toggle`	Control-Shift สลับภาษา
`grp:ctrl_alt_toggle`	Control-Alt สลับภาษา
`grp:alt_shift_toggle`	Alt-Shift สลับภาษา
`grp:menu_toggle`	Menu สลับภาษา
`grp:lwin_toggle`	Win ซ้าย สลับภาษา
`grp:rwin_toggle`	Win ขวา สลับภาษา
`grp:lshift_toggle`	Shift ซ้าย สลับภาษา
`grp:rshift_toggle`	Shift ขวา สลับภาษา
`grp:lctrl_toggle`	Ctrl ซ้าย สลับภาษา
`grp:rctrl_toggle`	Ctrl ขวา สลับภาษา

ตารางที่ 2.2: ตัวเลือก XKB สำหรับการแสดงสถานะด้วย LED

ตัวเลือก	ความหมาย
`grp_led:num`	ใช้ NumLock LED แสดงกลุ่มสองขึ้นไป
`grp_led:caps`	ใช้ CapsLock LED แสดงกลุ่มสองขึ้นไป
`grp_led:scroll`	ใช้ ScrollLock LED แสดงกลุ่มสองขึ้นไป

2.3 การตั้งค่า XIM

ดังที่ได้กล่าวไปแล้ว ว่า XIM เป็นส่วนจัดการเหตุการณ์ของการกดแป้นพิมพ์ โดยหลังจากเรียกใช้ XKB เพื่อแปลงค่า scan code เป็น keyboard symbol แล้ว XIM ก็จะแปลง keyboard symbol ให้เป็นรหัสอักขระตามโลแคลอีกต่อหนึ่ง ซึ่งในการแปลงนี้ แต่ละภาษาจะมีวิธีแปลงต่างกัน ภาษาในกลุ่ม CJK (จีน ญี่ปุ่น เกาหลี) จะมี XIM ที่ซับซ้อนเพราะตัวอักษรมีมากกว่าปุ่มที่มี จำเป็นต้องแปลงจากเสียงอ่านเป็นตัวอักษร ในขณะที่ XIM ของภาษาอังกฤษจะง่ายถึงกับแปลงแบบหนึ่งต่อหนึ่งได้เลย

ความแตกต่างดังกล่าว จึงทำให้ XIM กลายเป็นส่วนที่ทำงานตามโลแคล โดยเอกซ์วินโดว์ได้กำหนดให้ใช้หัวข้อ `LC_CTYPE` ของ POSIX locale เป็นตัวกำหนดเลือก XIM กล่าวคือ ถ้าเรากำหนด `LC_CTYPE` หรือเพียงแต่กำหนด `LANG` ให้เป็น `th_TH` ก็เท่ากับได้กำหนดให้ XIM ไทยทำงานแล้ว

อย่างไรก็ดี เอกซ์วินโดว์ยังอนุญาตให้มี XIM ได้หลายแบบในโลแคลเดียวกัน โดยอาศัย X locale modifiers ในการเลือก XIM สำหรับ XIM ไทยจะมีการกรองลำดับการป้อนอักขระ โดยแบ่งลำดับขั้นของความเคร่งครัดออกเป็น 3 ระดับ คือ

1. `Passthrough`: ไม่ตรวจสอบลำดับ
2. `BasicCheck`: ตรวจสอบลำดับแบบพื้นฐาน
3. `Strict`: ตรวจสอบลำดับแบบเคร่งครัดอักขรวิธี

การตั้ง X locale modifiers ทำได้โดยตั้งค่าตัวแปรระบบชื่อ `XMODIFIERS` โดย modifier ที่ใช้เลือก XIM คือ `"im"` โดยรูปแบบจะเป็น

```
export XMODIFIERS="@im=method"
```

เช่น `XMODIFIERS="@im=BasicCheck"` หมายถึงเลือกใช้การตรวจสอบลำดับแบบพื้นฐาน

หากไม่ระบุ `XMODIFIERS` เลย XIM ไทยจะเลือกใช้แบบ `BasicCheck` โดยปริยาย

2.4 ระบบป้อนอินพุตของ GTK+ 2

GTK+ 2 เป็น toolkit ที่เน้นเรื่องการสนับสนุนภาษาต่างๆ เป็นพิเศษ เริ่มตั้งแต่การสร้างไลบรารี Pango สำหรับแสดงข้อความภาษาต่างๆ ทั่วโลกขึ้นมาโดยเฉพาะ และอีกส่วนหนึ่งก็คือโครงสร้างของการป้อนข้อความภาษาต่างๆ โดยสามารถสร้างเป็นโมดูลต่างหากสำหรับแต่ละภาษาได้ เรียกว่า *GTK+ IM Module*

ใน GTK+ widget ต่างๆ ที่เกี่ยวกับการป้อนข้อความ เช่น text entry, text box จะมี context menu รายการหนึ่ง คือ “Input Methods” ซึ่งจะแสดงรายการของ GTK+ IM Module ที่ติดตั้งไว้ทั้งหมด ซึ่งปัจจุบัน ใน GTK+ 2 มี IM Module ที่ใช้ป้อนภาษาไทยได้คือ

- Default

- Thai (Broken)
- X Input Method

สองรายการแรก เป็นการแปลงรหัสแป้นพิมพ์เป็นอักขระภาษาไทยตรงๆ ไม่มีการตรวจสอบลำดับการพิมพ์ใดๆ ทั้งสิ้น ในส่วนของ “Thai (Broken)” นั้น เป็น IM Module ทดสอบที่ทางผู้พัฒนา GTK+ ใช้ทดลองโครงสร้างเท่านั้น ไม่ได้มุ่งหวังให้ใช้งานจริง ดังนั้น IM Module ที่น่าจะใช้งานได้ดีที่สุดในขณะนี้คือ “X Input Method” ซึ่งเป็นช่องทางเชื่อมไปยัง XIM ของ X Window นั่นเอง

สำหรับ IM Module ไทยสำหรับ GTK+ โดยเฉพาะนั้น กำลังอยู่ในระหว่างการพัฒนา

บทที่

3

การติดตั้งฟอนต์

3.1 ระบบฟอนต์บนเอกซ์วินโดว์

เอกซ์วินโดว์ (X Window) ถูกออกแบบไว้ให้ทำงานในระบบไคลเอนต์-เซิร์ฟเวอร์ (client-server) โดยแยกการทำงานออกเป็นสองส่วนหลักๆ คือ เอกซ์ไคลเอนต์ (X client) และเอกซ์เซิร์ฟเวอร์ (X server) โดยเอกซ์ไคลเอนต์ได้แก่โปรแกรมประยุกต์ต่างๆ ซึ่งจะขอใช้บริการเอกซ์เซิร์ฟเวอร์ในการติดต่อกับฮาร์ดแวร์ เช่น จอภาพ แป้นพิมพ์ เมาส์ ฯลฯ โดยที่ทั้งเอกซ์เซิร์ฟเวอร์และเอกซ์ไคลเอนต์ไม่จำเป็นต้องอยู่บนเครื่องเดียวกันเลยก็ได้ ทั้งสองส่วนสามารถติดต่อสื่อสารกันผ่านระบบเครือข่าย โดยปกติแล้วเอกซ์เซิร์ฟเวอร์จึงทำงานอยู่ที่เครื่องที่เรานั่งอยู่ที่หน้าจอ และเอกซ์ไคลเอนต์อาจอยู่ในเครื่องเดียวกัน หรือในเครื่องที่อยู่ไกลออกไปในระบบเครือข่ายก็ได้

ในการแสดงข้อความบนจอภาพ เอกซ์เซิร์ฟเวอร์จึงเป็นผู้จัดการเรื่องฟอนต์ต่างๆ โดย เอกซ์ไคลเอนต์สามารถขอดูรายชื่อฟอนต์ที่มีอยู่ทั้งหมด เลือกฟอนต์ที่ใช้ และสั่งเอกซ์เซิร์ฟเวอร์ให้วาดตัวอักษรบนจอภาพด้วยฟอนต์ที่กำหนดได้

อย่างไรก็ดี เมื่อเทียบกับงานส่วนอื่นๆ ของเอกซ์เซิร์ฟเวอร์ที่เป็นการควบคุมฮาร์ดแวร์โดยตรงแล้ว เรื่องของฟอนต์เป็นเรื่องที่ค่อนข้างเป็นอิสระจากฮาร์ดแวร์ ฟอนต์สำหรับเอกซ์เซิร์ฟเวอร์จึงสามารถแยกออกมาเป็นบริการต่างหากที่เรียกว่า เอกซ์ฟอนต์เซิร์ฟเวอร์ (X font server) หรือบางครั้งเรียกสั้นๆ ว่าฟอนต์เซิร์ฟเวอร์หรือ XFS ได้ ซึ่งจะทำให้เอกซ์เซิร์ฟเวอร์เฉพาะส่วนที่จัดการฮาร์ดแวร์มีขนาดเล็กลงจนอาจใส่ไว้ใน ROM อย่าง Network PC เลยก็ได้ และยังทำให้การเพิ่มและลบฟอนต์เป็นอิสระ สะดวกต่อการดูแลอีกด้วย รวมทั้งทำให้เป็นไปได้ที่จะใช้ฟอนต์ร่วมกันในเครือข่ายผ่านเซิร์ฟเวอร์เครื่องเดียวได้ด้วย

3.1.1 ระบบรายชื่อฟอนต์ของเอกซ์เซิร์ฟเวอร์

เอกซ์เซิร์ฟเวอร์จะรู้จักฟอนต์ต่างๆ จากรายชื่อพาธของฟอนต์ (font path) โดยสามารถขอดูหรือตั้งค่าได้ด้วยคำสั่ง `xset`

การดูค่าฟอนต์พาธของเอกซ์เซิร์ฟเวอร์ ใช้คำสั่ง

```
xset q
```

ซึ่งจะรายงานค่าต่างๆ ของเอกซ์เซิร์ฟเวอร์ โดยมี “Font Path” เป็นหัวข้อหนึ่ง ดังตัวอย่าง

```
thep@anubis:~$ xset q
...
Font Path:
  /usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/75dpi/
:unscaled,/usr/X11R6/lib/X11/fonts/100dpi/:unscaled,/usr/X11R6/l
ib/X11/fonts/Type1/,/usr/X11R6/lib/X11/fonts/TrueType/,/usr/X11R
6/lib/X11/fonts/75dpi/,/usr/X11R6/lib/X11/fonts/100dpi/
...
```

เป็นตัวอย่างของการใช้ฟอนต์จากเอกซ์เซิร์ฟเวอร์โดยตรง โดยเอกซ์เซิร์ฟเวอร์จะไปอ่านฟอนต์ในไดเรกทอรีต่างๆ ใน Font Path นี้มาใช้ทั้งหมด

การเพิ่มและลบฟอนต์พาธในเอกซ์เซิร์ฟเวอร์ ใช้คำสั่ง

```
xset {+fp|fp+|-fp|fp-} path[,path...]
```

คำสั่ง `xset +fp` และ `xset fp+` ใช้เพิ่มฟอนต์พาธไปที่ต้นและท้ายรายการของฟอนต์พาธ ตามลำดับ ส่วน `xset -fp` และ `xset fp-` ใช้ลบฟอนต์พาธที่กำหนดออกจากรายการฟอนต์พาธ

คำสั่ง

```
xset fp= path[,path...]
```

ใช้ตั้งค่าฟอนต์พาธให้เป็นค่าใหม่ที่กำหนด

คำสั่ง

```
xset fp default
```

ใช้ตั้งค่าฟอนต์พาธกลับเป็นค่าปริยายซึ่งตั้งไว้เป็นค่าเริ่มต้นที่ `/etc/X11/XF86Config-4` สำหรับ XFree86 รุ่น 4.0 ขึ้นไป หรือ `/etc/X11/XF86Config` สำหรับ XFree86 รุ่นเก่ากว่านั้น ใน `Section ‘‘Files’’` รายการ `FontPath`

อีกคำสั่งที่น่าสนใจคือ

```
xset fp rehash
```

เป็นคำสั่งที่สั่งให้เอกซ์เซิร์ฟเวอร์อ่านข้อมูลของฟอนต์ทั้งหมดใหม่อีกครั้ง ใช้เมื่อมีการแก้ไขรายการฟอนต์ในไดเรกทอรีใดไดเรกทอรีหนึ่ง

3.1.2 การใช้ฟอนต์ผ่านฟอนต์เซิร์ฟเวอร์

ดังได้กล่าวไปแล้วว่าเอกซ์เซิร์ฟเวอร์สามารถเรียกใช้ฟอนต์จากฟอนต์เซิร์ฟเวอร์ได้ โดยเพียงแต่ระบุในฟอนต์พาธของเอกซ์เซิร์ฟเวอร์ด้วยค่าที่แทนบริการของ XFS ซึ่งมีรูปแบบทั่วไปคือ

```
protocol/[host]:port
```

โดยที่

protocol หมายถึงโปรโตคอลที่บริการ XFS ใช้ เช่น `tcp` หรือ `unix` (Unix-domain socket)
host คือชื่อเครื่องที่ให้บริการ XFS
port คือหมายเลขพอร์ตที่รอรับบริการ

ซึ่งค่าเหล่านี้ ขึ้นอยู่กับค่าที่ตั้งไว้ที่ตัวบริการ XFS เอง ตัวอย่างเช่น
`tcp/font.mycompany.com:7100` → ใช้ TCP ที่พอร์ต 7100 ของเครื่อง font.mycompany.com
`tcp/localhost:7100` → ใช้ TCP ที่พอร์ต 7100 ของเครื่องเดียวกับเอกซ์เซิร์ฟเวอร์เอง
`unix/:7100` → ใช้ Unix-domain socket ที่พอร์ต 7100 ที่เครื่องเดียวกับเอกซ์เซิร์ฟเวอร์เอง

ส่วนที่ฟอนต์เซิร์ฟเวอร์เอง การตั้งค่าจะตั้งได้ที่ `/etc/X11/fs/config` ซึ่งสามารถกำหนดตัวเลือกของการเปิดบริการได้ ว่าจะให้ใช้ TCP หรือไม่ หรือจะเปลี่ยนพอร์ตจากค่าปกติ (คือ 7100) เป็นพอร์ตอื่น รวมทั้งระบุรายชื่อไดเรกทอรีทั้งหมดที่เก็บฟอนต์ได้ในหัวข้อ `catalogue`

โดยปกติแล้ว XFS จะเป็นบริการของระบบที่จะเปิดเมื่อบูตเครื่อง และสามารถสั่งปิดบริการหรือสั่งเริ่มบริการใหม่ได้เหมือนบริการระบบอื่นๆ โดยไฟล์ที่ใช้สั่งการโดยปกติจะอยู่ที่ `/etc/init.d/xfs` คุณสามารถเรียกไฟล์นี้ที่บรรทัดคำสั่งโดยใช้อาร์กิวเมนต์ `stop, start` หรือ `restart` ได้ เช่น เมื่อคุณแก้ค่าคอนฟิกต่างๆ ของ xfs แล้ว คุณก็สั่ง

```
/etc/init.d/xfs restart
```

เพื่อให้ xfs เริ่มทำงานใหม่จากค่าใหม่

การเพิ่มและลบฟอนต์ให้กับ XFS สามารถทำได้โดยแก้คอนฟิกแล้วเริ่มบริการ xfs ใหม่ได้ดังกล่าวข้างต้น หรือถ้าใช้ลินุกซ์ตระกูล RedHat หรือ Mandrake (รวมทั้งลินุกซ์ทะเล) ก็สามารถใช้คำสั่ง `/usr/sbin/chkfontpath` ได้ โดยไม่ต้องเริ่มบริการ xfs ใหม่ คำสั่ง `chkfontpath` มีรูปแบบเป็น

```
/usr/sbin/chkfontpath -l
/usr/sbin/chkfontpath [-a|-r] path
```

“`chkfontpath -l`” จะแสดงรายการไดเรกทอรีของฟอนต์ทั้งหมดที่ XFS ใช้อยู่ในปัจจุบัน ส่วน “`chkfontpath -a`” และ “`chkfontpath -r`” ใช้เพิ่มและลบไดเรกทอรีของฟอนต์ตามลำดับ

3.2 การติดตั้งฟอนต์บนเอกซ์วินโดว์

จากหัวข้อ 3.1 จะเห็นว่าจุดที่เราสามารถติดตั้งฟอนต์ได้จะมีสองจุด คือที่เอกซ์เซิร์ฟเวอร์โดยตรง หรือที่ฟอนต์เซิร์ฟเวอร์ก็ได้

โดยปกติแล้ว ในการใช้ลินุกซ์เดสก์ทอปโดยไม่ได้เชื่อมต่อระบบเครือข่ายโดยทั่วไป ทุกส่วนดังกล่าว ทั้งเอกซ์ไคลเอนต์, เอกซ์เซิร์ฟเวอร์ และฟอนต์เซิร์ฟเวอร์ก็จะอยู่ในเครื่องเดียวกัน แต่เดิมใน XFree86 3.3.6 นั้น การใช้ฟอนต์เซิร์ฟเวอร์จะมีข้อได้เปรียบตรงที่ฟอนต์เซิร์ฟเวอร์ ที่มีอยู่ในขณะนั้น เช่น xfs-ft, xfstt, xfs-xtt สามารถสนับสนุนฟอนต์ TrueType ที่ใช้กันในไมโครซอฟท์วินโดวส์ได้ ในขณะที่เอกซ์เซิร์ฟเวอร์จะใช้ได้เพียงฟอนต์บิตแมพและฟอนต์ Postscript เท่านั้น แต่ใน XFree86 4.0 ก็ได้เพิ่มการสนับสนุนฟอนต์ TrueType เข้าไว้ในเอกซ์เซิร์ฟเวอร์เองแล้ว ปัจจุบันจึงมีความนิยมใช้ฟอนต์ผ่านเอกซ์เซิร์ฟเวอร์โดยตรงมากพอๆ กับผ่านฟอนต์เซิร์ฟเวอร์

ดังที่ได้กล่าวไปแล้วในหัวข้อ 3.1 ถึงเรื่องระบบรายชื่อฟอนต์ของเอกซ์เซิร์ฟเวอร์และฟอนต์เซิร์ฟเวอร์ที่จะอ้างอิงเป็นไดเรกทอรี โดยที่ในแต่ละไดเรกทอรีต้องมีไฟล์ `fonts.dir` ระบุรายชื่อฟอนต์ทั้งหมดในไดเรกทอรีนั้นๆ ดังตัวอย่าง

```
8
thai18.pcf -nectec-fixed-medium-r-normal--18-180-72-72-c-90-tis620-0
thai18i.pcf -nectec-fixed-medium-i-normal--18-180-72-72-c-90-tis620-0
thai18b.pcf -nectec-fixed-bold-r-normal--18-180-72-72-c-90-tis620-0
thai18bi.pcf -nectec-fixed-bold-i-normal--18-180-72-72-c-90-tis620-0
thai7x18.pcf -thai-fixed-medium-r-normal--14-100-100-100-m-70-tis620-0
thai6x14.pcf -thai-fixed-medium-r-normal--10-100-75-75-c-60-tis620-0
thai8x20.pcf -thai-fixed-medium-r-normal--16-114-100-100-m-80-tis620-0
thai8x16.pcf -thai-fixed-medium-r-normal--16-114-100-100-p-100-tis620-0
```

บรรทัดแรกจะบอกว่าในไดเรกทอรีนี้มีฟอนต์ทั้งหมด 8 ฟอนต์ บรรทัดถัดมาอีก 8 บรรทัดเป็นรายการฟอนต์ดังกล่าว โดยคอลัมน์แรกเป็นชื่อไฟล์ที่เก็บฟอนต์ คอลัมน์ที่สองเป็นชื่อฟอนต์ที่เอกซ์วินโดว์ใช้เรียก

ไฟล์ `fonts.dir` โดยปกติจะสร้างด้วยคำสั่ง `mkfontdir` ซึ่งอาจต้องใช้ตัวเลือกเพิ่มเติมแล้วแต่ชนิดของฟอนต์ ดังจะได้กล่าวต่อไป

เมื่อเตรียมพร้อมไดเรกทอรีโดยสร้างไฟล์ `fonts.dir` ดังกล่าวแล้ว ก็สามารถลงทะเบียนไดเรกทอรีของเราเข้าไปยังเอกซ์เซิร์ฟเวอร์หรือฟอนต์เซิร์ฟเวอร์ได้ ดังได้กล่าวไปแล้วในหัวข้อ 3.1

3.2.1 การติดตั้งฟอนต์บิตแมพ

ฟอนต์บิตแมพสำหรับเอกซ์วินโดว์จะมีสองรูปแบบ คือ BDF และ PCF ฟอนต์ BDF จะมีรูปแบบเป็นไฟล์ข้อความ สามารถใช้เอดิเตอร์เปิดและแก้ไขได้ ส่วน PCF จะเป็นไฟล์ไบนารีที่คอมไพล์แล้ว ซึ่งจะมีขนาดเล็กกว่า

การติดตั้งฟอนต์บิตแมพถือว่าง่ายและตรงไปตรงมาที่สุดในบรรดาฟอนต์ทั้งหลาย คำสั่งที่ใช้สร้าง `fonts.dir` คือ

```
mkfontdir
```

ซึ่งจะสแกนหาฟอนต์ทั้งหมดในไดเรกทอรีแล้วดึงข้อมูลในฟอนต์มาสร้างเป็นรายการต่างๆ ใน `fonts.dir`

3.2.2 การติดตั้งฟอนต์ Type 1

ฟอนต์ Postscript Type 1 เป็นเทคโนโลยีฟอนต์เวกเตอร์จากค่าย Adobe และนิยมใช้มากในงานพิมพ์ ฟอนต์บนแมคอินทอชก็จะใช้ Type 1 เป็นหลัก และเอกซ์วินโดว์เองก็สนับสนุนฟอนต์ Type 1 มานานแล้ว

การสร้าง `fonts.dir` สำหรับฟอนต์เวกเตอร์ส่วนใหญ่หลังจากใช้โปรแกรมอัตโนมัติสแกนฟอนต์และสร้างเป็นรายการฟอนต์แล้ว อาจจะต้องปรับแก้อีกนิดหน่อย ดังนั้น โดยทั่วไปจึงไม่สร้างเป็น `fonts.dir` โดยตรงจากโปรแกรม แต่จะสร้างเป็นไฟล์ `fonts.scale` ซึ่งยังแก้ไขได้ และคำสั่ง `mkfontdir` จะมองหาไฟล์นี้เพื่อสร้างเป็น `fonts.dir` อีกที

คำสั่งสำหรับสแกนฟอนต์ Type 1 เพื่อสร้าง `fonts.scale` คือ

```
type1inst
```

แต่ถ้าคุณใช้ XFree86 4.3.0 เป็นต้นไป จะมีโปรแกรม `mkfontscale` ที่มีหลักการทำงานเหมือน `type1inst` คือสร้างไฟล์ `fonts.scale` สำหรับฟอนต์แบบเวกเตอร์ ซึ่งหมายถึงทั้ง Type 1 และ TrueType โดยสำหรับฟอนต์ภาษาไทย ควรระบุตัวเลือก `-e` เพื่อให้ค้นหาอักขระในชุดรหัสอักขระไทย ดังนี้

```
mkfontscale -e tis620-0 -e tis620-2
```

จากนั้น คำสั่งที่ใช้สร้าง `fonts.dir` จาก `fonts.scale` ก็คือ

```
mkfontdir
```

อนึ่ง สำหรับ XFree86 4.0 ขึ้นไป การใช้ฟอนต์ Type 1 จาก X server โดยตรง จะต้องโหลดโมดูล `type1` ด้วย โดยตั้งค่าได้ในไฟล์ `/etc/X11/XF86Config-4` ที่ `Section "Module"` เพิ่ม

```
Load "type1"
```

แต่ถ้าใช้ XFS ก็ไม่ต้องเซ็ตอะไรเพิ่ม

3.2.3 การติดตั้งฟอนต์ TrueType

ฟอนต์ TrueType เป็นฟอนต์เวกเตอร์ที่นิยมใช้มากในไมโครซอฟท์วินโดวส์ โดยเฉพาะในงานสร้างเอกสารแบบ What You See Is What You Get (WYSIWYG) สำหรับเอกซ์วินโดว์ ก็ได้มีความพยายามสนับสนุนฟอนต์ TrueType มาเป็นลำดับ ใน XFree86 3.3.6 นั้น ผู้ใช้ยังต้องใช้ฟอนต์เซิร์ฟเวอร์บางตัวที่ดัดแปลงให้ใช้ฟอนต์ TrueType ได้ เช่น xfsft, xfstt, xfs-xtt แต่ใน XFree86 รุ่น 4.0 ได้ผนวกเอาความสามารถดังกล่าวเข้าไว้ในเอกซ์เซิร์ฟเวอร์โดยตรง

ในไมโครซอฟท์วินโดวส์รุ่นแรกๆ (3.1) นั้น ฟอนต์ TrueType ทั้งหมดยังเป็น 8 บิตอยู่ จึงสามารถใช้กับรหัส มอก. 620 ของไทยได้โดยตรง แต่ในรุ่นถัดๆ มา ฟอนต์ TrueType ได้เปลี่ยนเป็น Unicode (16 บิต) หมด เพื่อเตรียมตัวสนับสนุนระบบหลากภาษาซึ่งต้องใช้อักขระจำนวนมาก การใช้ฟอนต์ TrueType ดังกล่าวในการแสดงผลอักขระ มอก. 620 จึงต้องมีกรรมวิธีในการแปลงรหัสเพิ่มเติมเล็กน้อย การติดตั้งฟอนต์ TrueType บนเอกซ์วินโดว์จึงต้องมีวิธีการพิเศษเล็กน้อย

เริ่มจากคำสั่งที่ใช้สร้าง `fonts.scale` จะใช้คำสั่ง

```
ttmkfdir > fonts.scale หรือ
ttmkfdir -e /usr/X11R6/lib/X11/fonts/encodings/encodings.dir
  -o fonts.scale
```

โดยในรูปแบบแรก ใช้กับ ttmkfdir รุ่นที่ 1 ที่แก้ไขให้รู้จักภาษาไทยแล้ว ได้แก่ ttmkfdir ใน Mandrake 8.0 ขึ้นไป, ใน Debian unstable version (sid), ในลินุกซ์ทะเล 4.0 ส่วนรูปแบบหลังใช้กับ ttmkfdir รุ่นที่ 2 เช่น ttmkfdir ใน RedHat 7.0 ขึ้นไป, ในลินุกซ์ทะเล 4.1

ttmkfdir จะสแกนฟอนต์ TrueType และสร้างรายการใน fonts.scale โดยดึงเอาทุกรหัสอักขระย่อยๆ ที่ตรวจพบออกมา รวมทั้ง tis620-0 ที่เราต้องการด้วย

อนึ่ง ใน XFree86 4.3.0 เป็นต้นไปจะมีโปรแกรม mkfontscale ที่มีหลักการทำงานเหมือน ttmkfdir คือสร้างไฟล์ fonts.scale สำหรับฟอนต์แบบเวกเตอร์ ซึ่งหมายถึงทั้ง Type 1 และ TrueType โดยสำหรับฟอนต์ภาษาไทย ควรระบุตัวเลือก -e เพื่อให้ค้นหาอักขระในชุดรหัสอักขระไทยดังนี้

```
mkfontscale -e tis620-0 -e tis620-2
```

นอกจากนี้ คำสั่ง mkfontdir สำหรับฟอนต์ TrueType ไทยก็ยังต้องใช้ตัวเลือกพิเศษ:

```
mkfontdir -e /usr/X11R6/lib/X11/fonts/encodings
```

ซึ่งนอกจากจะสร้าง fonts.dir ตามปกติแล้ว ยังสร้าง encodings.dir เพื่อให้ฟอนต์เซิร์ฟเวอร์หรือเอกซ์เซิร์ฟเวอร์ใช้ในการแปลงอักขระภาษาไทยให้เป็น Unicode เพื่อใช้ฟอนต์อีกด้วย

อนึ่ง สำหรับ XFree86 4.0 ขึ้นไป การใช้ฟอนต์ TrueType จากเอกซ์เซิร์ฟเวอร์โดยตรง จะต้องโหลดโมดูล freetype ด้วย โดยตั้งค่าได้ในไฟล์ /etc/X11/XF86Config-4 ที่ Section "Module" เพิ่ม

```
Load "freetype"
```

แต่ถ้าใช้ฟอนต์เซิร์ฟเวอร์ก็ไม่ต้องเซ็ตอะไรเพิ่ม

3.3 ฟอนต์ที่ฝั่งไคลเอนต์

XFree86 ตั้งแต่รุ่น 4.0 ขึ้นไป ได้มีส่วนขยายเพื่อการแสดงผลแบบลบรอยหยัก (anti-aliasing) ที่เรียกว่า X Render โดยอาศัยการเกลี่ยสีในบริเวณรอบๆ เส้น ทำให้เส้นขอบรูปดูเรียบขึ้นเมื่อมองด้วยตาเปล่า และยังมีไลบรารี Xft เพื่อเรียกใช้ส่วนขยาย X Render ในการวาดตัวอักษรจากฟอนต์ TrueType โดยอาศัยความสามารถของไลบรารี FreeType อีกด้วย

ระบบฟอนต์ที่ Xft ใช้ จะเป็นฟอนต์ที่ฝั่งเอกซ์ไคลเอนต์ ระบบติดตั้งฟอนต์จึงไม่ได้รวมอยู่กับ X server หรือ XFS ทั้งนี้จะเป็นประโยชน์สำหรับระบบ WYSIWYG ที่จำเป็นต้องใช้ฟอนต์สำหรับแสดงผลและฟอนต์สำหรับการพิมพ์ออกเครื่องพิมพ์ร่วมกัน

ใน XFree86 รุ่นก่อน 4.3.0 จะติดตั้งฟอนต์ Xft ผ่านไฟล์ /etc/X11/XftConfig การเพิ่มฟอนต์ TrueType ใหม่ให้กับ Xft ก็เพียงแต่เพิ่มบรรทัด

```
dir "path"
```

เมื่อ *path* เป็นชื่อพาธเต็มที่เก็บฟอนต์ จากนั้นสั่ง

```
xftcache
```

เพื่อปรับปรุงรายการไฟล์

อย่างไรก็ดี หากคุณใช้ XFree86 ตั้งแต่รุ่น 4.3.0 เป็นต้นไป (เช่น Red Hat 8.0 เป็นต้นไป) Xft ที่ใช้จะเป็น Xft2 ซึ่งได้แยกส่วนจัดการฟอนต์ออกมาเป็นไลบรารี fontconfig ต่างหาก ซึ่งสามารถใช้ร่วมกับ toolkit และโปรแกรมต่างๆ ที่ไม่ได้ทำงานบน X Window ได้ด้วย ไฟล์ `/etc/X11/XftConfig` จึงถูกเลิกใช้งาน และเปลี่ยนไปใช้ไฟล์ `/etc/fonts/fonts.conf` ของ fontconfig แทน

โดยปกติ `/etc/fonts/fonts.conf` จะเก็บรากของฟอนต์ระดับบนสุดเท่านั้น โดยฟอนต์ทั้งหมดที่อยู่ใต้รากนั้นลงไปจะถูกดึงขึ้นมาใช้ทั้งหมด ค่าปกติของรากในไฟล์ดังกล่าวได้แก่

```
<fontconfig>
  ...
  <dir>/usr/X11R6/lib/X11/fonts</dir>
  <dir>/usr/share/fonts</dir>
  <dir>~/.fonts</dir>
  ...
</fontconfig>
```

ดังนั้น หากคุณติดตั้งฟอนต์ไว้ภายใต้รากเหล่านี้ คุณก็ไม่จำเป็นต้องแก้ไฟล์ `fonts.conf` ดังกล่าว และโดยปกติแล้ว ไลบรารี fontconfig ก็จะ update รายการฟอนต์เป็นระยะๆ ทุกครั้งที่มีการใช้ฟอนต์อยู่แล้ว แต่คุณก็ยังสามารถสั่ง update รายการฟอนต์ทันทีที่ติดตั้งฟอนต์เสร็จ โดยใช้คำสั่ง

```
fc-cache -f[v] [dir]
```

ตัวเลือก `-f` (force) เป็นการบังคับให้สร้าง cache ใหม่โดยไม่ต้องตรวจสอบเวลาของไฟล์ต่างๆ ตัวเลือก `-v` (verbose) เป็นการสั่งให้แสดงขั้นตอนการทำงานด้วย ส่วน *dir* คือชื่อไดเรกทอรีที่จะ update โดยถ้าไม่กำหนดจะเป็นการ update ใหม่ทุกราก

ไฟล์ cache ที่ `fc-cache` สร้าง คือ `fonts.cache-1` ซึ่งจะประจำอยู่ในทุกไดเรกทอรีย่อย คุณสามารถขอดูรายการฟอนต์ทั้งหมดที่ fontconfig รู้จักได้โดยสั่ง

```
fc-list
```

ปัจจุบัน ทั้ง GTK+ 2 และ Qt 3 ได้เปลี่ยนมาใช้ fontconfig หมดแล้ว ดังนั้น การติดตั้งฟอนต์ผ่าน fontconfig จึงมีผลต่อโปรแกรมต่างๆ บน GNOME 2 และ KDE 3 ทันที

บทที่

4

การพิมพ์เอกสารภาษาไทย

4.1 ระบบการพิมพ์บนลินุกซ์

ระบบการพิมพ์บนลินุกซ์ในปัจจุบันค่อนข้างหลากหลาย มีทั้งแบบที่อ้างอิงระบบ spool แบบยูนิกซ์ดั้งเดิม และแบบที่ไม่มีการเข้าคิว ในการใช้งานทั่วไป เรามักจะเลือกระบบ spool เพราะเหมาะกับการทำงานแบบ multi-task, multi-user และโดยเฉพาะการแชร์เครื่องพิมพ์ในระบบเครือข่าย ระบบ spool ในยูนิกซ์ดั้งเดิมคือ LPD ของ BSD แต่ด้วยขีดจำกัดต่างๆ ทำให้ปัจจุบันไม่เป็นที่นิยมใช้ แต่จะใช้ระบบ LPRng (LPR-new generation) ที่พัฒนาขึ้นใหม่ตามโปรโตคอลเดิม และอีกระบบหนึ่งที่ได้รับความนิยมมากขึ้นเรื่อยๆ คือ CUPS (Common UNIX Printing System) ซึ่งนอกจากจะสนับสนุนโปรโตคอลของ LPD แบบเดิม ยังสนับสนุน Internet Printing Protocol (IPP) อีกด้วย

ในระบบ spooling ของ LPD จะมี daemon `lpd` (line printer daemon) ที่คอยให้บริการรับพิมพ์งานรออยู่ เมื่อผู้ใช้ส่งงานพิมพ์ด้วยคำสั่ง `lpr` ก็จะเป็นการติดต่อไปยัง `lpd` เพื่อเอางานเข้าคิวไว้ และทันทีที่เครื่องพิมพ์ว่าง ก็จะหยิบงานออกจากคิวออกมาพิมพ์

ในขั้นตอนการพิมพ์งานนี้เอง คิวแต่ละคิวในระบบ `lpd` ซึ่งดูแลเครื่องพิมพ์แต่ละเครื่องสามารถกำหนดให้ส่งงานผ่านตัวกรอง (filter) ก่อนส่งข้อมูลให้เครื่องพิมพ์ได้ ตัวกรองนี้นี่เอง ที่ช่วยแปลงข้อมูลให้เป็นคำสั่งเฉพาะของเครื่องพิมพ์แต่ละชนิด

ในระบบยูนิกซ์ ภาษาหลักที่ใช้สั่งงานเครื่องพิมพ์ก็คือ Postscript และโปรแกรมต่างๆ ก็จะพิมพ์งานโดยจัดหน้าด้วยภาษา Postscript เป็นหลัก ดังนั้น เครื่องพิมพ์ที่สนับสนุน Postscript จึงใช้กับยูนิกซ์ได้ทันที แต่เครื่องพิมพ์ Postscript ก็มักมีราคาแพง เครื่องพิมพ์เล็กๆ ที่ใช้ตามบ้าน โดยเฉพาะเครื่อง Inkjet มักใช้ภาษาอื่น การสนับสนุนเครื่องพิมพ์เหล่านั้น ก็คือการติดตั้งตัวกรองที่แปลงข้อมูล Postscript ให้เป็นรูปแบบที่เครื่องพิมพ์นั้นๆ รู้จักนั่นเอง

โปรแกรมที่ทำหน้าที่สำคัญนี้บนลินุกซ์ก็คือ Ghostscript ซึ่งผลิตโดยบริษัท Aladdin Ghostscript เป็นชุดซอฟต์แวร์ interpreter สำหรับภาษา Postscript ซึ่งเป็นกลไกภายในให้กับระบบการพิมพ์ โปรแกรม `gv` (ghostview) ที่มาพร้อมกับ Ghostscript เป็นโปรแกรมหนึ่งที่ใช้ Ghostscript ตีความเพื่อแสดงตัวอย่าง (preview) ไฟล์ Postscript บนจอภาพ นอกจากนี้ Ghostscript ยังรวมตัวกรอง

ข้อมูลซึ่งแปลงข้อมูล Postscript ให้เป็นรูปแบบเฉพาะของเครื่องพิมพ์ด้วย

ทั้งหมดนั้น เป็นโครงสร้างที่ทำให้เกิดระบบการพิมพ์ที่ใช้ Postscript เป็นภาษากลาง อย่างไรก็ดี สำหรับการพิมพ์ไฟล์ต่างๆ ที่ไม่ใช่ Postscript เช่น ไฟล์รูปภาพ ไฟล์ข้อความ ก็มีตัวกรองอยู่จำนวนหนึ่งที่สามารถแปลงข้อมูลให้เป็น Postscript ก่อนส่งพิมพ์ได้ เช่น `a2ps` แต่เมื่อรวมกับปัญหาอื่นๆ ที่ต้องจัดการ ทำให้มีผู้พัฒนาตัวกรองขึ้นมาเพื่อใช้กับ print spooler ซึ่งจะจัดการทุกอย่างอย่างครบวงจร ตั้งแต่การแปลงรูปแบบไฟล์ การสั่งคำสั่งพิเศษสำหรับเครื่องพิมพ์ตามแต่ความสามารถของเครื่องพิมพ์ (เช่น การเลือกขนาดกระดาษ การเลือกถาดกระดาษ การเลือกความละเอียด ฯลฯ) ไปจนถึงการแปลงข้อมูลให้เป็นรูปแบบที่เครื่องพิมพ์รู้จัก (ผ่าน Ghostscript) ตัวอย่างของระบบดังกล่าวก็คือ foomatic

foomatic จะรวบรวมฐานข้อมูลของเครื่องพิมพ์ชนิดต่างๆ เพื่อสร้างเป็นไฟล์ PPD (Postscript Printer Description) โดยอัตโนมัติ ไฟล์ PPD ใช้เป็นข้อมูลเฉพาะของเครื่องพิมพ์ชนิดต่างๆ ในการสั่งการแบบ Postscript โดยปกติจะมีมากับแผ่น driver ของเครื่องพิมพ์ คุณสามารถใช้ไฟล์ตรงนั้น หรือจะสร้างโดยอัตโนมัติจากฐานข้อมูลของ foomatic ก็ได้ สิ่งที่ foomatic เตรียมให้อีกก็คือตัวกรองสำหรับแปลงรูปแบบข้อมูล (โดยเรียกใช้โปรแกรมประเภท `a2ps` อีกต่อหนึ่ง) และสั่งพิมพ์แบบพิเศษ

4.2 Ghostscript

จะเห็นว่า หัวใจสำคัญจุดหนึ่งของการพิมพ์ก็คือ Ghostscript ซึ่งจะช่วยจัดการงานพิมพ์ในขั้นสุดท้ายได้ไฟล์ Postscript ที่สร้างมาจากโปรแกรมต่างๆ นั้น ถ้าเราสนใจในเรื่องการพิมพ์ข้อความภาษาไทยก็คงจะแบ่งได้เป็นสองจำพวกหลักๆ คือประเภทที่ฝังฟอนต์ (embed font) กับไม่ฝังฟอนต์

สำหรับเอกสารประเภทที่ฝังฟอนต์มาในข้อมูล Postscript จะไม่มีปัญหาในการพิมพ์ เพราะ Ghostscript หรือเครื่องพิมพ์ (ที่สนับสนุน Postscript) สามารถใช้ข้อมูลฟอนต์ที่ฝังมาในการสร้างรูปภาพตัวอักขระได้ ตัวอย่างโปรแกรมที่ให้ผลลัพธ์แบบนี้ได้แก่ LaTeX, ระบบการพิมพ์ของ GNOME 2 และ KDE 3, OpenOffice.org แต่สำหรับเอกสารอีกประเภทหนึ่งที่ไม่ฝังฟอนต์ เช่น ผลลัพธ์จาก AbiWord 2 (AbiWord 1.x ยังฝังฟอนต์อยู่ แต่ผู้เขียนเข้าใจว่าเป็นเพราะ AbiWord 2 กำลังอยู่ในระหว่างการย้ายระบบจาก GNOME 1.4 มายัง GNOME 2 จึงอาจมีบางส่วนยังไม่สมบูรณ์) หรือจาก MagicPoint (โปรแกรม presentation จากญี่ปุ่น) จำเป็นต้องติดตั้งฟอนต์ในสารบบของ Ghostscript ด้วย เพื่อที่ Ghostscript จะสามารถดึงฟอนต์มาวาดอักขระก่อนส่งให้เครื่องพิมพ์ได้

ไฟล์ที่เก็บ font map ของ Ghostscript คือ `/usr/share/gs/`*`version`*`/Fontmap` เมื่อ *`version`* คือเวอร์ชันของ Ghostscript เนื้อหาในไฟล์ดังกล่าวจะมีสองประเภท คือการกำหนดฟอนต์จริง (real font) กับการกำหนดฉายาฟอนต์ (font alias) ตัวอย่างเช่น คุณต้องการติดตั้งฟอนต์ชุด Norasi ให้กับ Ghostscript คุณก็กำหนดฟอนต์จริง โดยใช้วงเล็บครอบชื่อไฟล์ (Ghostscript 5 ขึ้นไปสนับสนุนไฟล์ `.ttf` แล้ว)

```
/Norasi             (norasi_n.ttf) ;
/Norasi-Bold        (norasi_b.ttf) ;
/Norasi-Italic      (norasi_i.ttf) ;
/Norasi-BoldItalic  (norasi_bi.ttf) ;
```

โดยคุณอาจกำหนดฉายาฟอนต์ได้ เช่น คุณต้องการให้เอกสารที่อ้างฟอนต์ Times โดยไม่ฝังฟอนต์มาใช้ Norasi แทน ก็ทำได้โดยเพิ่มหรือแก้บรรทัดต่อไปนี้

```
/Times-Roman        /Norasi ;
/Times-Italic       /Norasi-Italc ;
/Times-Bold         /Norasi-Bold ;
/Times-BoldItalic   /Norasi-BoldItalic ;
```

หากคุณใช้ Debian คุณจะพบว่า ระบบ defoma (Debian Font Management) ได้จัดการเรื่องนี้ให้แล้วโดยอัตโนมัติเมื่อติดตั้งฟอนต์จากแหล่งของ Debian

4.3 Mozilla กับ Xprint

ระบบการพิมพ์บนยูนิกซ์เริ่มมีความเปลี่ยนแปลงอย่างมากเมื่อลักษณะการใช้งานในลักษณะเดสก์ทอปแบบบูรณาการเริ่มพัฒนาเป็นรูปเป็นร่าง จากระบบเดิมที่แต่ละโปรแกรมต่างคนต่างพิมพ์โดยมาบรรจบกันที่ภาษา Postscript และระบบ print spooling ก็เริ่มมีการใช้โครงสร้างร่วมกันในระดับที่สูงขึ้น ไม่ว่าจะผ่าน toolkit (GTK+ หรือ Qt) หรือผ่านระบบการพิมพ์โดยเฉพาะ (gnome-print)

ในขณะเดียวกัน ประเด็นนี้ก็ถูกยกขึ้นมาในระบบ X Window เกิดเป็นส่วนขยาย Xprint ขึ้นมา ซึ่ง Xprint จะใช้โครงสร้างเหมือน X server ทุกประการ แต่เปลี่ยนการวาดจากติดต่อ graphic card มาเป็นการสร้างเอกสารด้วยภาษาจัดหน้า (Page Description Language) ซึ่งปัจจุบันสนับสนุนสองภาษา คือ Postscript และ PCL

แต่การพิมพ์ลงบนกระดาษก็ไม่เหมือนกับการวาดบนจอภาพอย่างที่ X server ทำ เพราะต้องมีการแบ่งหน้า Xprint server จึงให้บริการส่วนขยาย Xp เพิ่มเติมด้วย โดย client จะติดต่อกับ Xprint server ผ่าน *Print Context* ด้วยไลบรารี libXp

การพิมพ์ใน Mozilla ก็เริ่มใช้ Xprint มาได้ระยะหนึ่งแล้ว แต่ Xprint ที่มีมาใน XFree86 ยังไม่สมบูรณ์ จึงต้องใช้ Xprint ฉบับที่ใช้กับ Mozilla โดยเฉพาะ ซึ่งดาวน์โหลดได้จาก `xprint.mozdev.org`

การใช้ Xprint กับ Mozilla และโปรแกรมอื่นๆ ก็มีขั้นตอนหลักๆ 3 ขั้น คือ 1. การตั้งค่า Xprint server 2. การ start Xprint server (`Xprt`) 3. การตั้งค่า client เราจะมาดูขั้นตอนต่างๆ อย่างคร่าวๆ

4.3.1 การตั้งค่า Xprint server

ไดเรกทอรีสำหรับตั้งค่า Xprint โดยปกติจะอยู่ใต้ราก `/usr/X11R6/lib/X11/xserver/` ยกเว้น Debian จะย้ายไปอยู่ที่ราก `/usr/share/Xprint/xserver/` ในการกล่าวถึงรากนี้ต่อไปในเอกสารนี้ จะขอเรียกเป็นตัวแปรว่า `$XPCONFIGDIR`

ไฟล์ที่น่าสนใจสำหรับการตั้งค่า Xprint ได้แก่

`$XPCONFIGDIR/C/print/Xprinters` ลิสต์รายการเครื่องพิมพ์ที่ Xprint จะใช้ โดยปกติ `Xprt` จะใช้เครื่องพิมพ์ในรายการนี้ บวกกับเครื่องพิมพ์ที่ติดตั้งไว้ใน print spool ทั้งหมด แต่ก็สามารถห้ามการเพิ่มรายการเครื่องพิมพ์จาก spool ได้โดยกำหนดบรรทัด `Augment_Printer_List` โดยตรง

`$XPCONFIGDIR/C/print/attributes/printer` กำหนดแบบหรือรุ่นของเครื่องพิมพ์ต่างๆ ในระบบ โดยปกติ หากเครื่องพิมพ์ของคุณสนับสนุน Postscript (ด้วยความช่วยเหลือของ Ghostscript หรือ foomatic ก็ตามแต่) ก็จะใช้แบบ `PSdefault`

`$XPCONFIGDIR/C/print/attributes/document` กำหนดค่าต่างๆ ของการพิมพ์เอกสาร เช่น ขนาดกระดาษ จำนวนชุดที่จะพิมพ์ ความละเอียด เป็นต้น

`$XPCONFIGDIR/C/print/attributes/job` กำหนดการทำงานของแต่ละ job ที่สั่งพิมพ์ เช่น กำหนดให้ส่ง mail ถึงผู้ใช้เมื่อพิมพ์เสร็จ

`$XPCONFIGDIR/C/print/models/{PSdefault...}` เก็บพารามิเตอร์และฟอนต์ของเครื่องพิมพ์แบบต่างๆ โดยเฉพาะไฟล์ 'model-config' ซึ่งบรรยายความสามารถต่างๆ ที่เครื่องพิมพ์สนับสนุน เช่น ขนาดกระดาษ ถาดกระดาษ ความละเอียด โหมดพิมพ์ ฯลฯ

การปรับแต่งอย่างง่ายที่เป็นไปได้แบบหนึ่งก็คือ เซ็ตเครื่องพิมพ์ใน print spool ให้สนับสนุน Postscript (เพราะโปรแกรมอื่นๆ บนลินุกซ์มักจะใช้ Postscript อยู่แล้ว) จากนั้น ค่าปกติของ X-print จะเลือกใช้เครื่องพิมพ์แบบ `PSdefault` อยู่แล้ว คุณอาจจำเป็นต้องปรับความละเอียดของการพิมพ์ที่ไฟล์ `attributes/document` ดังกล่าวข้างต้นถ้าผลการพิมพ์ออกมาไม่เต็มหน้ากระดาษ โดยอ้างอิงค่าต่างๆ ที่สนับสนุนทั้งหมดจากไฟล์ `models/PSdefault/model-config`

4.3.2 การ start Xprint server

ตัวโปรแกรม Xprint server คือ `Xprt` ซึ่งสามารถ start ได้ทั้งแบบที่เป็นบริการของระบบและแบบที่เป็นโปรเซสของผู้ใช้ (ตามแบบฉบับของ X server ทั่วไป)

การ start สำหรับระบบ

โดยปกติ Xprint จะมีสคริปต์สำหรับเริ่ม/จบบริการตามแบบ System V ที่ `/etc/init.d/xprint` มาให้อยู่แล้ว คุณเพียงแต่สั่งเหมือนๆ กับบริการทั่วไป

```
/etc/init.d/xprint {start|stop|restart}
```

การ start สำหรับผู้ใช้

ผู้ใช้สามารถสั่งเริ่ม `Xprt` ได้ง่ายๆ โดยสั่ง

```
Xprt -fp /usr/X11R6/lib/X11/fonts/TrueType,/usr/X11R6/lib/X11/fon
ts/misc :12
```

ค่าในตัวเลือก `-fp` ได้แก่รายการของพาธที่อยู่ของฟอนต์ที่คุณจะเลือกใช้ ข้อควรระวังก็คือ อย่าลืมเติม `/usr/X11R6/lib/X11/fonts/misc` ด้วยเสมอ มิฉะนั้น `Xprt` จะหาฟอนต์ 'fixed' ไม่พบ และไม่ยอมทำงาน ส่วนค่าสุดท้าย ':12' คือหมายเลข display ของ Xprint server

การตั้งค่า client

คล้ายๆ กับที่ X client ทั่วไปสามารถอ้างอิงถึง X server ผ่านตัวแปรระบบ `DISPLAY` โดยกำหนดให้เป็นชื่อเครื่องและหมายเลข display และ screen ที่ต้องการ (เช่น `export DISPLAY=:0.0` คือเครื่องเดียวกัน ที่ display หมายเลข 0 และ screen หมายเลข 0) การอ้างอิงถึง Xprint server ก็จะ

อ้างผ่านตัวแปรระบบ XPSERVERLIST โดยระบุ display ของ Xprt ที่กำหนดขณะ start เช่น ถ้าคุณ start Xprt เช่นนี้

```
Xprt -fp /usr/X11R6/lib/X11/fonts/TrueType,/usr/X11R6/lib/X11/fon
ts/misc :12
```

คุณก็กำหนดตัวแปร XPSERVERLIST ดังนี้

```
export XPSERVERLIST=:12
```

คำสั่งง่ายๆ ที่จะตรวจสอบว่าระบบของคุณสามารถติดต่อกับ Xprt ที่คุณ start ไว้ได้ ก็โดยใช้คำสั่ง

```
thep@anubis:~$ xplsprinters
printer: c82@:12
printer: spooldir_tmp_Xprintjobs@:12
```

ซึ่งจะแสดงรายการเครื่องพิมพ์ทั้งหมดที่คุณสามารถติดต่อผ่าน Xprint ได้

สังเกตว่า ชื่อเครื่องพิมพ์ใน Xprint จะมีรูปแบบ

```
printer@display
```

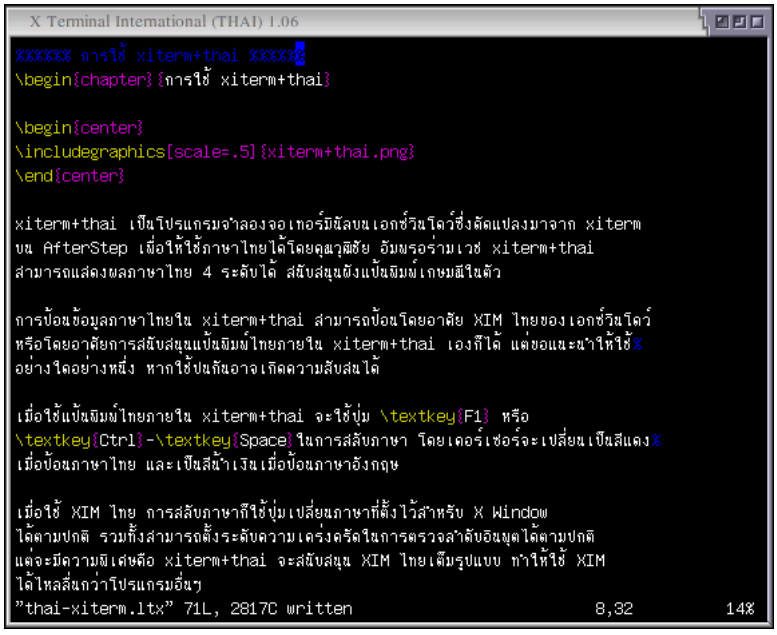
โดยที่ *printer* เป็นชื่อเครื่องพิมพ์ของ lpd หรือเครื่องพิมพ์ที่คุณกำหนดในไฟล์ Xprinters ส่วน *display* ก็คือชื่อ display ของ Xprt

หากเซ็ตค่าต่างๆ เหล่านี้ถูกต้อง เมื่อสั่งพิมพ์หน้าเว็บใน Mozilla ก็จะมีรายชื่อเครื่องพิมพ์ของ X-print ให้เลือกใน print dialog และหากคุณระบุ font path โดยรวมฟอนต์ไทยให้กับ Xprt ขณะ start ด้วย คุณก็จะสามารถพิมพ์เว็บภาษาไทยได้!

บทที่

5

การใช้ xiterm+thai



xiterm+thai เป็นโปรแกรมจำลองจอเทอร์มินัลบนเอกซ์วินโดว์ซึ่งดัดแปลงมาจาก xiterm บน AfterStep เพื่อให้ใช้ภาษาไทยได้โดยคุณวุฒิชัย อัมพรอร่ามเวช xiterm+thai สามารถแสดงผลภาษาไทย 4 ระดับได้ สนับสนุนผังแป้นพิมพ์เกษมณีในตัว

การป้อนข้อมูลภาษาไทยใน xiterm+thai สามารถป้อนโดยอาศัย XIM ไทยของเอกซ์วินโดว์ หรือโดยอาศัยการสนับสนุนแป้นพิมพ์ไทยภายใน xiterm+thai เองก็ได้ แต่ขอแนะนำให้ใช้อย่างใดอย่างหนึ่ง หากใช้ปนกันอาจเกิดความสับสนได้

เมื่อใช้แป้นพิมพ์ไทยภายใน xiterm+thai จะใช้ปุ่ม F1 หรือ Ctrl-Space ในการสลับ

ภาษา โดยเคอร์เซอร์จะเปลี่ยนเป็นสีแดงเมื่อป้อนภาษาไทย และเป็นสีน้ำเงินเมื่อป้อนภาษาอังกฤษ

เมื่อใช้ XIM ไทย การสลับภาษาก็ใช้ปุ่มเปลี่ยนภาษาที่ตั้งไว้สำหรับ X Window ได้ตามปกติ รวมทั้งสามารถตั้งระดับความเคร่งครัดในการตรวจลำดับอินพุตได้ตามปกติ แต่จะมีความพิเศษคือ xiterm+thai จะสนับสนุน XIM ไทยเต็มรูปแบบ ทำให้ใช้ XIM ได้ไหลลื่นกว่าโปรแกรมอื่นๆ

อย่างไรก็ตาม การใช้แป้นพิมพ์ไทยภายในจะไม่มีการตรวจลำดับอินพุตใดๆ

xiterm+thai มีตัวเลือกในการเรียกใช้ที่เกี่ยวกับภาษาไทยที่น่าสนใจคือ

- `-tspace` *n*: เริ่มชดเชยช่องว่างให้กับสระบน-ล่างเมื่อพบช่องว่าง *n* ช่อง
- `-tkb` *mode*: เลือกผังแป้นพิมพ์
  - `ket` = เกษมณี
  - `tis` = มอก. 820-2533
- `-tim` *mode*: เลือกระดับการตรวจลำดับอินพุตด้วย XIM
  - `Passthrough` = ไม่ตรวจ
  - `BasicCheck` = ตรวจขั้นพื้นฐาน
  - `Strict` = ตรวจเคร่งครัด

นอกจากนี้ ยังสามารถตั้งค่าเริ่มต้นเพื่อใช้โดยอัตโนมัติทุกครั้งที่เรียกได้ โดยกำหนดไว้ที่ไฟล์ `~/.Xdefaults` (สำหรับ RedHat, Mandrake, ลินุกซ์ทะเล) หรือ `~/.Xresources` (สำหรับ Debian) โดยมีรูปแบบเป็น

```
xiterm*resource: value
```

โดยที่ resource ต่างๆ ที่ตั้งค่าได้ ได้แก่

- `thai_space:` *n* เหมือนตัวเลือก `-tspace`
- `thai_keyboard:` *mode* เหมือนตัวเลือก `-tkb`
- `thai_im:` *mode* เหมือนตัวเลือก `-tim`
- `cursorColor:` *color* สีของเคอร์เซอร์เมื่ออยู่ในโหมดภาษาอังกฤษของผังแป้นพิมพ์ภายใน
- `cursorColorThai:` *color* สีของเคอร์เซอร์เมื่ออยู่ในโหมดภาษาไทยของผังแป้นพิมพ์ภายใน

บทที่

6

การใช้ภาษาไทยใน LaTeX

LaTeX เป็นโปรแกรมเรียงพิมพ์ (typesetter) ที่ได้รับความนิยมมากในหมู่นักวิชาการ เนื่องจากความสามารถในการเขียนสูตรคณิตศาสตร์ การใช้สัญลักษณ์พิเศษ เช่น เครื่องหมายทางสัทศาสตร์ (phonetic symbol) รวมทั้งความสามารถในการทำ cross reference ระหว่างเนื้อหาส่วนต่างๆ การทำดรรชนี การทำสารบัญอัตโนมัติ จึงมีการใช้ LaTeX ในการเขียนบทความวิชาการและในการสร้างหนังสืออย่างกว้างขวาง

LaTeX *ไม่ใช่*โปรแกรมประมวลคำ (word processor) ในลักษณะ What You See Is What You Get (WYSIWYG) แต่เป็นการคอมไพล์จาก markup language ให้เป็นรูปแบบที่แสดงผลทางอุปกรณ์ต่างๆ เช่น จอภาพ หรือเครื่องพิมพ์ได้ การเตรียมเอกสาร LaTeX จึงสามารถใช้โปรแกรมเอดิเตอร์ธรรมดาได้เลย

ในเอกสารนี้จะไม่ขออธิบายละเอียดถึง markup ต่างๆ ของ LaTeX ผู้อ่านสามารถศึกษาเพิ่มเติมจากหนังสือที่อธิบายเรื่องนี้โดยเฉพาะ แต่จะกล่าวถึงเฉพาะส่วนที่เกี่ยวกับการใช้ภาษาไทยในเอกสาร LaTeX เท่านั้น

6.1 การใช้ภาษาไทยด้วย babel package

babel เป็นแพคเกจเพิ่มเติมที่ทำให้ LaTeX สามารถใช้ภาษาต่างๆ นอกเหนือจากภาษาอังกฤษได้ โดยสนับสนุนการตั้งค่าฟอนต์สำหรับแต่ละภาษา ข้อความที่ใช้เรียกส่วนต่างๆ ของเอกสาร เช่น หัวเรื่องบทคัดย่อ บทที่ รูปภาพ ตาราง ฯลฯ รวมทั้งข้อกำหนดอื่นๆ เฉพาะสำหรับภาษานั้นๆ ที่อาจจะมี

ด้วยแพคเกจ thailatex ที่พัฒนาโดยเนคเทคร่วมกับกลุ่มทำงานลินุกซ์ไทย เราสามารถใช้ภาษาไทยโดยอาศัย babel นี้ได้ โดยใช้ตัวเลือก `thai` กับ preamble `\usepackage{babel}` ในเอกสารดังนี้:

```
\usepackage[thai]{babel}
```

ตัวอย่างเช่น

```
\documentclass[a4paper]{article}
\usepackage[thai]{babel}

\begin{document}
สวัสดี ชาวโลก
\end{document}
```

ความจริงแล้ว แพคเกจ babel สามารถเลือกใช้ภาษาได้มากกว่าหนึ่งภาษา โดยใช้จุลภาค (,) คั่นระหว่างตัวเลือกต่างๆ โดย babel จะถือเอาภาษาสุดท้ายเป็นภาษาหลักของเอกสาร และเมื่อจะเปลี่ยนภาษาก็ใช้คำสั่ง `\selectlanguage{}` หรือ `\latintext` กับ `\thaitext` เช่น

```
\documentclass[a4paper]{article}
\usepackage[thai,english]{babel}

\begin{document}
English people say ‘Hello,’ while Thai say
‘{\selectlanguage{thai} สวัสดี}.’

\latintext English people say ‘Hello,’ while Thai say
‘{\thaitext  สวัสดี}.’
\end{document}
```

จะใช้ฟอนต์และคำเรียกส่วนต่างๆ ของเอกสารเป็นภาษาอังกฤษเป็นหลัก โดยสามารถแทรกข้อความภาษาไทยในเอกสารได้

นอกจากนี้ thailatex ตั้งแต่รุ่น 0.2.5 ขึ้นไป ยังมีตัวเลือก `thainum` เพิ่มเติมสำหรับ babel เพื่อใช้เลขไทยในที่ต่างๆ เช่น เลขบท เลขหน้า เลขหัวข้อย่อย รูปที่ ตารางที่ ฯลฯ อีกด้วย

6.2 การตัดคำด้วย swath และ cttex

thailatex ได้กำหนดคำสั่ง `\wbr` ให้ใช้เป็นจุดตัดคำ ซึ่งหากเราขยันพอที่จะใส่คำสั่ง `\wbr` เพื่อจัดเอกสารด้วยตัวเองก็ย่อมได้ แต่ก็เป็นเรื่องที่กินแรงมิใช่น้อย จึงได้มีโปรแกรมแบ่งคำอัตโนมัติมาช่วย

โปรแกรมแบ่งคำที่สร้างมาเพื่อใช้กับ thailatex โดยเฉพาะคือ swath โดยมีวิธีใช้ดังนี้:

```
swath -f latex < input-file > output-file
```

อย่างไรก็ดี โปรแกรมที่สร้างมาก่อน swath คือ cttex ก็ยังสามารถใช้กับ thailatex ที่ใช้ `\wbr` ได้ โดยกำหนดอาร์กิวเมนต์ให้กับ cttex ดังนี้:

```
cttex -W < input-file > output-file
```

6.3 การคอมไพล์เอกสาร LaTeX

ตามปกติแล้ว การคอมไพล์เอกสาร LaTeX ธรรมดาที่ไม่มีดรรชนีหรือส่วนขยายใดๆ จนถึงขั้นออกเครื่องพิมพ์นั้น มีสองขั้นตอน คือ

1. คอมไพล์ไฟล์ LaTeX ให้เป็น `.dvi` (device independent) โดยใช้คำสั่ง `latex`
2. แปลงไฟล์ `.dvi` เป็นไฟล์ `.ps` (PostScript) ซึ่งพิมพ์ออกเครื่องพิมพ์ได้ โดยใช้คำสั่ง `dvips`

อย่างไรก็ดี สำหรับเอกสารที่ใช้ภาษาไทยเป็นหลัก โดยปกติจะมีอีกขั้นตอนหนึ่งมาก่อนขั้นตอนแรก คือการผ่านโปรแกรมแบ่งคำอัตโนมัติ เช่น swath หรือ cttex

และหากต้องการสร้างเอกสารเป็น PDF (Portable Document Format) ก็ทำได้โดยใช้โปรแกรม `ps2pdf` แปลงไฟล์ `.ps` ที่ได้ให้เป็น `.pdf`

ดังนั้น จากไฟล์ LaTeX ภาษาไทย เช่น `sample.tex` ตัวอย่างของคำสั่งสำหรับคอมไพล์ตามลำดับจึงได้แก่

```
swath -f latex < sample.tex > sample.ttex
latex sample.ttex
latex sample.ttex #  cross reference
dvips -o sample.ps sample.dvi
ps2pdf sample.ps #  PDF output
```

บทที่

7

การใช้ภาษาไทยใน LyX

LyX เป็นเอดิเตอร์แบบ GUI ที่ช่วยอำนวยความสะดวกในการสร้างเอกสาร LaTeX โดยใช้ฟอนต์แสดงผลบนจอภาพให้ผู้ใช้ได้เห็นความแตกต่างของเนื้อความส่วนต่างๆ ที่ถูก markup ไว้ รวมทั้งมีเมนูไดอะล็อก และกล่องเครื่องมือสำหรับช่วย markup ทำให้ผู้ใช้ไม่ต้องจำคำสั่ง LaTeX ด้วย

การใช้ภาษาไทยใน LyX นอกจากจะต้องมี thailatex ที่ใช้งานได้แล้ว ก็จำเป็นต้องตั้งค่าต่างๆ เป็นพิเศษจากภาษาอังกฤษบ้าง แต่ก็ไม่มากมายอะไรนัก

ผู้เขียนขอขอบคุณ คุณชนพ ศิลปอนันต์ สำหรับข้อมูลต่างๆ ในบทนี้

7.1 การตั้งค่าฟอนต์สำหรับแสดงผล

เรียกเมนู

Edit → Preferences

เลือกแท็บ

Look&Feel → Screen Fonts

ป้อนชื่อฟอนต์สำหรับแสดงข้อความแบบ Roman, Sans Serif และ Typewriter ฟอนต์เหล่านี้จะใช้ในการแสดงผลบนจอภาพเท่านั้น ไม่ได้ใช้ในการพิมพ์ และไม่จำเป็นต้องเป็นฟอนต์เดียวกับฟอนต์ที่ใช้พิมพ์ เพราะ LyX ไม่ได้เป็น WYSIWYG อยู่แล้ว

อนึ่ง ในช่อง Encoding ของฟอนต์ ควรป้อนเป็น `tis620-0`

7.2 การตั้งค่าแป้นพิมพ์

เรียกเมนู

Edit → Preferences

เลือกแท็บ

Lang Opts → Language

เลือกเอา Keybord map ป้อน 1st เป็น null และ 2nd เป็น thai-kedmanee

ออกจากโปรแกรมและแก้ไขไฟล์ `~/.lyx/preference` โดยเพิ่มบรรทัด

```
\bind "C-backslash"  "keymap-toggle"
```

ซึ่งจะเป็นการตั้งค่าให้ใช้ `Ctrl-Backslash` ในการสลับภาษาไทย-อังกฤษ

7.3 การสร้างเอกสารภาษาไทยด้วย LyX

เมื่อสร้างเอกสารใหม่ ให้เลือกเมนู

Layout → Document

จะได้ไดอะล็อกสำหรับตั้งค่าสไตล์ของเอกสารว่าเป็น `article` หรือ `book` หรืออื่นๆ หลังจากตั้งค่าตามต้องการแล้ว ให้เลือกแท็บ Language

ภายใต้แท็บ Language ที่หัวข้อ Language เลือก `thai` และที่หัวข้อ Encoding เลือก `default`

บทที่

8

การใช้ภาษาไทยใน text console

ด้วยความสามารถของการ์ด EGA/VGA ในการโปรแกรม character generator ได้ ทำให้แม้ในโหมดข้อความ ก็สามารถแสดงอักขระที่ไม่ใช่ภาษาอังกฤษได้ รวมทั้งภาษาไทยด้วย เพียงแต่จะไม่สามารถจัดระดับได้เท่านั้น เพราะต้องแสดงผลหนึ่งอักขระต่อหนึ่งคอลัมน์ตามปกติ

เคอร์เนลลินุกซ์ก็สนับสนุนความสามารถนี้ พร้อมทั้งการตั้งค่าผังแป้นพิมพ์ใหม่ด้วย โดยในการสั่งการ จะใช้ชุดโปรแกรมที่ชื่อ console-tool และจะมีชุด console-data สำหรับรวบรวมข้อมูลฟอนต์และผังแป้นพิมพ์ภาษาต่างๆ แยกต่างหาก

รากที่เก็บข้อมูลของ console-data จะแตกต่างกันไปในลินุกซ์แต่ละตัว เช่น

- Linux TLE/Red Hat: `/lib/kbd`
- Mandrake: `/usr/lib/kbd`
- Debian: `/usr/share`
- Slackware: `/usr/share`

ในที่นี้จึงขอเรียกรากดังกล่าวด้วยตัวแปร `$LCTROOT` โดยขอให้ผู้อ่านแทนค่าในใจตามชนิดของลินุกซ์ที่ใช้อยู่

8.1 การติดตั้งฟอนต์

ฟอนต์สำหรับคอนโซลจะเป็นไฟล์ชนิด `.psf` (PC Screen Font) และแหล่งที่เก็บฟอนต์จะอยู่ที่ `$LCTROOT/consolefonts`

โดยปกติแล้ว ฟอนต์ PSF จะเป็นฟอนต์ 8 บิต แต่ในไดรเวอร์จอของเคอร์เนลรุ่นใหม่ๆ จะอ้างอักขระแบบยูนิโค้ด 16 บิตแล้ว จึงต้องมีตารางเพิ่มอีกหนึ่งตารางสำหรับแปลงค่าอักขระยูนิโค้ดจากไดรเวอร์ให้เป็น glyph เรียกว่า **Screen Font Map (SFM)** แต่ฟอนต์ PSF ก็อนุญาตให้ฝัง SFM ลงในฟอนต์ได้เช่นกัน ซึ่งสามารถตรวจสอบได้ด้วยคำสั่ง

```
psfgettable font.psf
```

ซึ่งจะแสดงแมพดังกล่าวถ้ามี และในการติดตั้งฟอนต์ที่มี SFM ฝังอยู่แล้ว ก็ไม่จำเป็นต้องใช้ SFM ภายนอกอีก

อีกประการหนึ่งที่ต้องคำนึงก็คือ โหมดการทำงานของคอนโซลจะมีสองโหมด คือ UTF mode และ byte mode โดยใน UTF mode ไดรเวอร์จอจะตีความอักขระที่โปรแกรมส่งมาให้แสดงผลเป็นลำดับของการเข้ารหัส UTF-8 แล้วตีความออกมาเป็นยูนิโค้ดโดยตรง แต่ถ้าเป็น byte mode ก็จะต้องมีตารางแปลงค่ารหัสอักขระ 8 บิตที่โปรแกรมส่งมาให้เป็นยูนิโค้ดก่อนแสดงผล (ย้ำอีกทีว่าไดรเวอร์จอของเคอร์เนลจะอ้างอักขระแบบยูนิโค้ดเสมอ) ตารางนี้เรียกว่า Application Charset Map (ACM) ซึ่งโดยปกติสำหรับภาษาไทยจะต้องใช้ เพราะเรายังใช้โลแคลที่เป็นรหัส TIS-620 อยู่

ทั้งแมพ SFM และ ACM จะเก็บไว้ที่ `$LCTROOT/consoletrans`

การโหลดฟอนต์, SFM และ ACM ดังกล่าว จะใช้คำสั่ง `consolechars` โดยมีรูปแบบการเรียกดังนี้

```
consolechars [-m acm] [-f font] [-u sfm]
```

โดยที่

acm ได้แก่ชื่อของไฟล์ ACM ที่เก็บไว้ที่ `$LCTROOT/consoletrans` โดยไม่ต้องระบุนามสกุล `.acm` หรือ `.acm.gz`

sfm ได้แก่ชื่อของไฟล์ SFM ที่เก็บไว้ที่ `$LCTROOT/consoletrans` โดยไม่ต้องระบุนามสกุล `.sfm` หรือ `.sfm.gz`

font ได้แก่ชื่อของไฟล์ PSF ที่เก็บไว้ที่ `$LCTROOT/consolefonts` โดยไม่ต้องระบุนามสกุล `.psf` หรือ `.psf.gz`

ตัวอย่างเช่น

```
consolechars -m tis620 -f tis-phaisarn.f16
```

จะโหลดฟอนต์ `$LCTROOT/consolefonts/tis-phaisarn.f16.psf.gz` และ ACM จาก `$LCTROOT/consoletrans/tis620.acm.gz` ให้กับไดรเวอร์จอ

8.2 การตั้งค่าแป้นพิมพ์

ผังแป้นพิมพ์ของ console-tools จะเก็บไว้ที่ `$LCTROOT/keymaps/i386` โดยแยกไดเรกทอรีย่อยตามผัง ผังแป้นพิมพ์ของไทย (ในลินุกซ์ทะเล 4.1) จะเก็บไว้ที่ `$LCTROOT/keymaps/i386/qwerty`

การโหลดผังแป้นพิมพ์ให้กับไดรเวอร์ของเคอร์เนล ใช้คำสั่ง `loadkeys` โดยมีรูปแบบคำสั่งดังนี้

```
loadkeys map
```

โดยที่

map ได้แก่ชื่อของไฟล์ kmap ที่เก็บไว้ที่ `$LCTROOT/keymaps/i386/*` โดยไม่ต้องระบุนามสกุล `.kmap` หรือ `.kmap.gz`

ตัวอย่างเช่น

```
loadkeys th-tis-38.tis620
```

จะโหลดผังแป้นพิมพ์ `$LCTROOT/keymaps/i386/qwerty/th-tis-38.tis620.kmap.gz` ให้กับไดรเวอร์แป้นพิมพ์

8.3 การตั้งค่าเริ่มต้นอัตโนมัติ

สำหรับลินุกซ์ทะเลและ Red Hat สามารถตั้งค่าเริ่มต้นของฟอนต์และผังแป้นพิมพ์ของคอนโซลของระบบได้ที่ไฟล์สองไฟล์คือ

- `/etc/sysconfig/i18n` โดยตั้งค่าตัวแปร
  - `SYSFONT` สำหรับฟอนต์
  - `UNIMAP` สำหรับ SFM
  - `SYSFONTACM` สำหรับ ACM
- `/etc/sysconfig/keyboard` โดยตั้งค่าตัวแปร
  - `KEYTABLE` สำหรับผังแป้นพิมพ์

8.4 แหล่งดาวน์โหลด

เนื่องจากการสนับสนุนภาษาไทยใน console-tools นี้ ยังอยู่ในขั้นทดลองใช้งาน ยังไม่ได้ check-in เข้าไปที่โครงการต้นน้ำ ผู้ที่ไม่ได้ใช้ลินุกซ์ทะเลอาจจะตั้งค่าตามที่กล่าวมานี้ไม่ได้ จึงขอให้ดาวน์โหลดข้อมูลและชุดโปรแกรม console-tools ที่แก้ไขแล้วได้ที่

http://linux.thai.net/sf_alpha/thai-console

บทที่

9

การแปลงรหัสข้อมูลภาษาไทย

9.1 การแปลงรหัสข้อมูลด้วย iconv

ในยุคหัวเลี้ยวหัวต่อของการย้ายระบบไปใช้ยูนิโค้ด (Unicode) แทนรหัสท้องถิ่นเพื่อความสะดวกในการสนับสนุนระบบหลากภาษา (multilingual) อันจะทำให้ซอฟต์แวร์สามารถสนับสนุนภาษาต่างๆ ในโลกนี้ได้อย่างมีระบบระเบียบยิ่งขึ้นนี้ ย่อมจะมีหลายครั้งหลายคราวที่ต้องมีการแปลงข้อมูลไปมาระหว่างรหัสอักขระท้องถิ่นกับยูนิโค้ด หรือแม้แต่แปลงไปมาระหว่างรหัสอักขระท้องถิ่นด้วยกันเองที่มีหลากหลายระบบ

เครื่องมือแปลงรหัสตัวหนึ่งที่ใช้ได้สะดวกมากๆ ที่เกิดขึ้นมาตามธรรมชาติของการใช้โลแคลก็คือ iconv

ในการลงทะเบียนโลแคลใหม่ใน GNU C library แต่ละโลแคลนั้น สิ่งหนึ่งที่จะต้องบรรยายอยู่แล้วก็คือ รหัสอักขระที่ใช้ในโลแคลนั้นๆ พร้อมผังที่เรียกว่า repertoire map ที่บอกว่า อักขระแต่ละตัวที่ใช้นั้น อยู่ที่ตำแหน่งใดในตารางยูนิโค้ด ดังนั้น iconv จึงเพียงแต่อาศัยข้อมูลรหัสอักขระที่มีพร้อมอยู่แล้วเหล่านี้

หากจะดูรายการรหัสอักขระทั้งหมดที่ iconv รู้จัก ก็สามารถเรียกดูได้โดยสั่ง

```
iconv -l
```

ส่วนการแปลงรหัสอักขระนั้น ก็มีรูปแบบการสั่งงานคือ

```
iconv -f from-code -t to-code [inputfile] [-o outputfile]
```

โดยหากไม่ระบุ *inputfile* หรือ *outputfile* ก็จะหมายความถึง standard input และ standard output ตามลำดับ ตัวอย่างเช่น คำสั่งต่อไปนี้ใช้แปลงไฟล์ `input.txt` จากรหัส มอก. 620 (TIS-620) ไปเป็นยูนิโค้ดในรูปแบบ UTF-8 โดยเขียนผลลัพธ์ที่ `output.txt`:

```
iconv -f TIS-620 -t UTF-8 input.txt -o output.txt
```

หรือจะใช้ร่วมกับ pipe ก็ได้ เช่น

```
gzip -cd web_utf8.html.gz | iconv -f UTF-8 -t TIS-620 | more
```