บทที่ 4

การพัฒนาโปรแกรมฟิงเกอร์พลัส

จากหลักการของการขยายสมรรถนะของโปรแกรมฟิงเกอร์ ที่ได้กล่าวไปในบทที่แล้ว สำหรับในบทนี้ เป็นการกล่าวถึง เครื่องมือ และวิธีการต่าง ๆ ในการพัฒนาโปรแกรม รวมไปถึงรายละเอียดของส่วนต่าง ๆ และตัวอย่างโปรแกรมบริการที่ได้พัฒนาขึ้นด้วย

โครงสร้างหลักของฟิงเกอร์พลัส

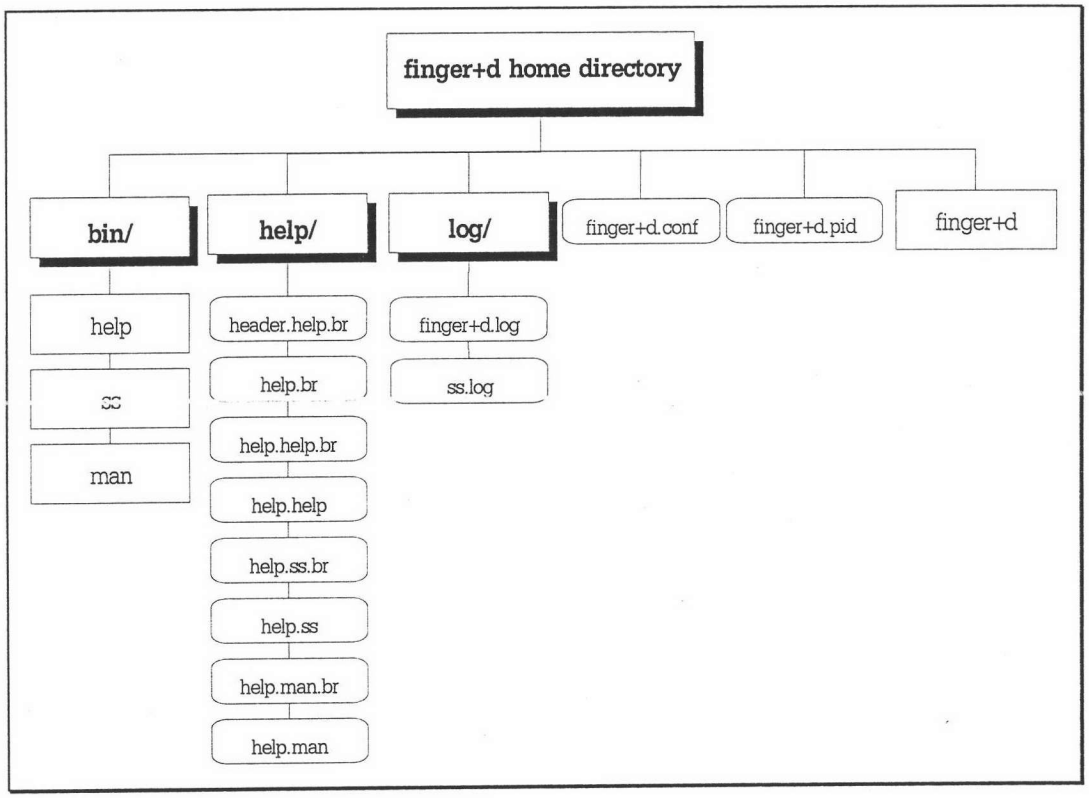
ชุดโปรแกรมฟิงเกอร์พลัสประกอบด้วยโปรแกรมฟิงเกอร์พลัส โปรแกรมบริการต่าง ๆ และแฟ้มข้อมูลที่เกี่ยวข้อง รวมกันอยู่ในไดเรกทอรี่ย่อยหนึ่ง เรียกว่า ไดเรกทอรีบ้าน (home directory) ของฟิงเกอร์พลัส มีลักษณะการจัดแฟ้มข้อมูลต่าง ๆ ในไดเรกทอรีย่อยดังรูปที่ 4.1

สำหรับรายละเอียดของแฟ้มข้อมูลต่าง ๆ ดูได้จากภาคผนวก ก

รูปแบบของการสอบถามไปยังฟิงเกอร์พลัส

จากข้อจำกัดด้านความสามารถในการรับส่งข้อมูลของระบบฟิงเกอร์แบบเดิม ระหว่างฟิงเกอร์ไคลเอ็นต์ และฟิงเกอร์เซอร์ฟเวอร์ จึงต้องมีการออกแบบรูปแบบของการสอบถามขึ้นใหม่ ซึ่งเดิมนั้น คำขอที่ฟิงเกอร์ไคลเอ็นต์สามารถส่งไปให้ ฟิงเกอร์เซอร์ฟเวอร์ได้ เป็นเพียงชื่อผู้ใช้ที่ต้องการให้ค้นหาข้อมูล และ/หรือ พารามิเตอร์ l (แอล) แทนความหมายถึงว่า ต้องการข้อมูลของผู้ใช้คนดังกล่าวแบบสมบูรณ์ ดังรูปแบบ

finger [-l] [username]@hostname



รูปที่ 4.1 แสดงโครงสร้างของแฟ้มข้อมูลต่าง ๆ ในระบบฟิงเกอร์พลัส

พบว่า รูปแบบของการสอบถามแบบเดิม ถ้านำมาใช้กับระบบฟิงเกอร์พลัส จะไม่สามารถเพิ่มเติมส่วนของคำขอที่ให้ระบุชื่อของโปรแกรมบริการ และอาร์กิวเมนต์ ที่ต้องการส่งมาให้กับทั้งโปรแกรมบริการ และตัวโปรแกรม finger+d ได้ ดังนั้นการออกแบบรูปแบบของคำขอใหม่ จึงต้องออกแบบให้ครอบคลุมความต้องการต่อไปนี้

1. ต้องสามารถใช้โปรแกรมฟิงเกอร์ไคลเอ็นต์เดิม ในการส่งคำขอนี้ไปยังเครื่องที่มี finger+d รอรับอยู่ได้
2. ต้องมีความเข้ากันได้กับรูปแบบของการสอบถามเดิมของฟิงเกอร์โปรโตคอล
3. ต้องสามารถสื่อสารถึง ข้อมูลตามความต้องการทั้งหมดของผู้ใช้ ไปยังเครื่องที่มี finger+d รอรับอยู่ ได้อย่างครบถ้วนใน 1 ครั้งของการสอบถาม

4. ต้องไม่ใช้อักขระพิเศษในระบบยูนิกซ์ ที่จะทำให้การตีความหมายโดยเชลล์ของระบบยูนิกซ์ (unix shell) ผิดพลาดไปจากที่ผู้ใช้ต้องการ

จากหลักการของระบบฟิงเกอร์พลัส ที่ให้ส่วนประมวลผลเสริมหน้าเป็นส่วนที่รับคำขอจากไคลเอ็นต์ แล้วทำการตัดสินใจว่าต้องส่งต่อคำขอให้กับโปรแกรมบริการใด สามารถสรุปได้ว่า รูปแบบคำขอของระบบฟิงเกอร์พลัส ต้องประกอบด้วยส่วนต่าง ๆ ดังต่อไปนี้

1. ส่วนที่ระบุว่าเป็นคำขอมายังระบบฟิงเกอร์พลัส หรือเป็นคำขอที่มีมายังระบบฟิงเกอร์แบบเดิม

2. ส่วนที่เป็นอาร์กิวเมนต์ไปให้กับตัวโปรแกรม finger+d เอง เพื่อควบคุมพฤติกรรมการแสดงผลลัพธ์กลับสู่ไคลเอ็นต์

3. ส่วนที่ระบุว่าต้องการให้ finger+d เรียกใช้โปรแกรมบริการใดให้

4. ส่วนที่เป็นอาร์กิวเมนต์เพื่อส่งไปให้กับตัวโปรแกรมบริการที่ระบุในส่วนที่ 3

โดยที่แต่ละส่วนต้องถูกส่งผ่านระบบเครือข่ายไปด้วยกัน ยังเครื่องที่ระบุที่มีโปรแกรม finger+d รอรับอยู่ และถูกแยกจากกันโดยโปรแกรม finger+d

ดังนั้นจึงออกแบบให้รูปแบบคำขอข้อมูล มีส่วนประกอบต่าง ๆ ดังรูปแบบต่อไปนี้

finger [.[argument1][.application [.argument2]][@hostname]]

โดยมีข้อกำหนดคือ

1. คำขอข้อมูลที่ขึ้นต้นด้วยเครื่องหมายจุด (.) ถือว่าเป็นคำขอที่มีมายังโปรแกรม finger+d โดยตรง ถ้าต้องการสอบถามข้อมูลโดยผ่านระบบฟิงเกอร์โปรโตคอลแบบเดิม จะต้องไม่ขึ้นต้นคำขอข้อมูลด้วยเครื่องหมายจุด

2. แต่ละส่วนของคำขอข้อมูล ถูกแยกออกจากกันด้วยเครื่องหมายจุดเช่นกัน และแม้ว่าคำขอข้อมูลส่วนใดจะละเลย หรือไม่ต้องการใส่ ก็ต้องคงเครื่องหมายจุดไว้ด้วย

3. คำขอข้อมูลทั้งหมด ต้องติดกันโดยตลอด ไม่มีการเว้นวรรคด้วยช่องว่าง

4. ถ้าเป็นคำขอข้อมูลที่ขึ้นต้นด้วยเครื่องหมายจุด ต้องตามด้วย อาร์กิวเมนต์1 (argument1) คือ อาร์กิวเมนต์ที่ต้องการส่งให้กับตัวโปรแกรม finger+d เอง เพื่อให้มีผลควบคุมผลลัพธ์ที่จะส่งกลับไปยังไคลเอ็นต์นั่นเอง โดยที่อาร์กิวเมนต์นี้อาจละเลยไม่ใส่ได้ แต่ถ้าใส่ต้องเป็นอักษรภาษาอังกฤษ หรือเครื่องหมายพิเศษ ตั้งแต่ 1 ตัวอักษรขึ้นไป มาประกอบกัน อาร์กิวเมนต์1 ที่สามารถใช้ได้ให้ ดูจากภาคผนวก ข

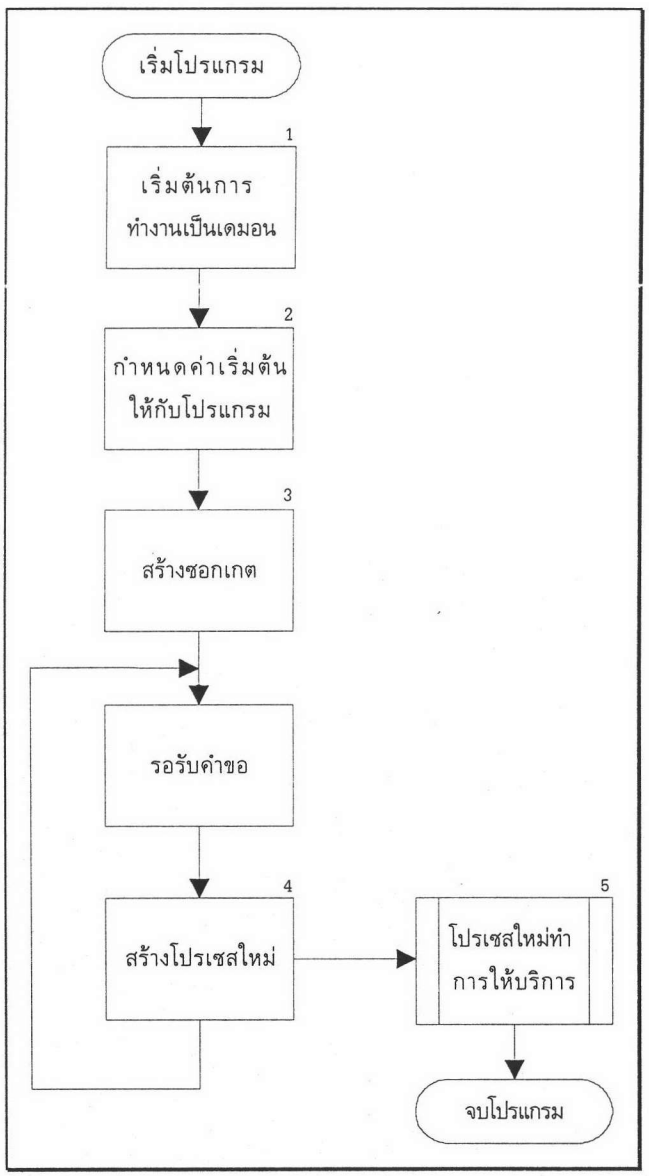
5. หลังจากอาร์กิวเมนต์1 จะเป็นชื่อของโปรแกรมบริการที่ต้องการเรียกใช้ สำหรับวิธีเรียกดูรายชื่อของโปรแกรมบริการที่มี สามารถดูได้จากภาคผนวก ข

6. ส่วนสุดท้ายของคำขอข้อมูลเป็น อาร์กิวเมนต์2 คือ ส่วนที่เป็นอาร์กิวเมนต์ที่ต้องการส่งให้กับตัวโปรแกรมบริการที่ระบุ โดยที่ส่วนนี้สามารถมีซ้ำ ๆ กันได้ และให้คั่นด้วยเครื่องหมายจุดเช่นกัน

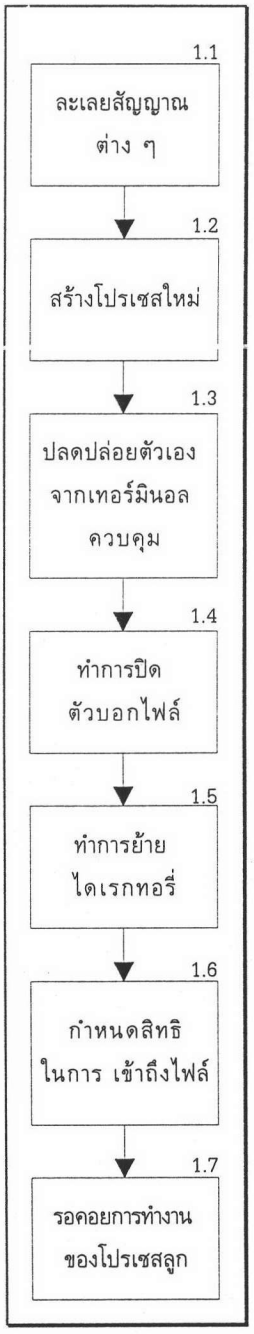
7. ต่อจากส่วนสุดท้ายของคำขอข้อมูล จะเป็นส่วนของชื่อเครื่องที่มีโปรแกรม finger+d ทำงานรอรับคำขอข้อมูลดังกล่าวอยู่ โดยมีข้อกำหนดของส่วนนี้เช่นเดียวกับที่กำหนดในฟิงเกอร์โปรโตคอลแบบเดิม โปรแกรมฟิงเกอร์ไคลเอ็นต์จะเป็นผู้ตรวจสอบในส่วนนี้ว่าจะต้องส่งคำขอส่วนต้นทั้งหมด ไปยังเครื่องใด และส่วนนี้จะไม่ถูกส่งไปกับคำขอข้อมูล

ส่วนประมวลผลเสริมหน้า

สำหรับส่วนประมวลผลเสริมหน้านี้ มีขั้นตอนการทำงานเป็นลำดับดังแสดงในรูปที่ 4.2 และจะได้กล่าวถึงชุดคำสั่งที่เกี่ยวข้องในแต่ละขั้นตอนต่อไป



รูปที่ 4.2 แสดงขั้นตอนการทำงานของ finger+d



รูปที่ 4.3 แสดงขั้นตอนการเริ่มต้นทำงานให้เป็นเดมอน

1. การเริ่มต้นการทำงานเป็นเดมอน

ในการทำงานของโปรแกรม finger+d ซึ่งมีลักษณะการทำงานเป็นเซอร์ฟเวอร์โปรแกรม ต้องมีการทำงานอยู่ตลอดเวลา แม้ว่าผู้ใช้ที่สั่งให้โปรแกรมนี้ทำงาน จะออกจากการใช้งานระบบไปแล้วก็ตาม เรียกการทำงานของโปรแกรมในลักษณะนี้ว่า โปรเซสที่ทำงานเบื้องหลัง หรือ แบ็คกราวน์โปรเซส (background process) และเรียกโปรแกรมที่ทำงานในลักษณะนี้ว่า เดมอนโปรแกรม (daemon program) การที่โปรแกรมสามารถทำงานเป็นเดมอนโปรแกรมได้นั้น มีขั้นตอนดังแสดงในรูปที่ 4.3 และมีรายละเอียดดังต่อไปนี้

1.1. ต้องทำการละเลยต่อสัญญานต่าง ๆ ที่ส่งจากระบบมายังเดมอนโปรแกรม เมื่อเดมอนโปรแกรมต้องการรับหรือส่งข้อมูลออกทางเทอร์มินัล เนื่องจากว่า ในการทำงานเป็นเดมอนโปรแกรมนั้น ตัวเดมอนโปรแกรมต้องปลดปล่อยตัวเองออกจากเทอร์มินัลที่ผู้ใช้ทำการสั่งให้เดมอนโปรแกรมนี้ทำงาน ซึ่งเรียกว่า เทอร์มินัลควบคุม (control terminal) จึงทำให้เดมอนโปรแกรมไม่สามารถสื่อสารข้อมูลใด ๆ ผ่านเทอร์มินัลได้ แต่ถ้าเมื่อใดตัวเดมอนต้องการส่งข่าวสารใด ๆ ออกไปยังเทอร์มินัล หรือต้องการรับข้อมูลเข้าผ่านทางเทอร์มินัล ระบบจะทำการส่งสัญญาน (signal) มายังเดมอนโปรแกรมว่าไม่สามารถติดต่อกับเทอร์มินัลได้ ซึ่งสัญญานดังกล่าวนี้จะทำให้เดมอนโปรแกรมหยุดทำงาน จึงต้องทำการละเลยต่อสัญญานเหล่านี้ (Stevens 1991,p. 75) โดยใช้ ฟังก์ชั่นจากไลบรารีฟังก์ชั่น (library function) คือ signal() ดังนี้

```
#include  <signal.h>
void signal (SIGNAL, SIG_IGN);
```

โดยที่อาร์กิวเมนต์ตัวแรกคือ ชื่อของสัญญาน ที่ต้องการให้ละเลย และ SIG_IGN เป็นส่วนที่ระบุให้ทำการละเลยต่อสัญญานดังกล่าว

1.2. ทำการสร้างโปรเซสลูก หรือฟอร์ค (fork) เพื่อให้โปรเซสลูกทำงานต่อไปในลักษณะเบื้องหลัง แล้วโปรเซสเดิมก็เลิกการทำงานไป โดยใช้ ซิสเต็มคอล (system call) คือ fork() ดังนี้

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

โดยที่ซิสเต็มคอล fork() นี้จะให้ค่าหมายเลขกระบวนการของโปรเซสลูกแก่โปรเซสแม่ และจะให้ค่า 0 แก่โปรเซสลูก ซึ่งสามารถใช้ค่านี้ในการตรวจสอบได้ว่าโปรเซสดังกล่าวเป็นโปรเซสแม่หรือโปรเซสลูก

1.3. ทำการปลดปล่อยตัวเดมอนโปรแกรมเองออกจากเทอร์มินัลควบคุม โดยการกำหนดให้ตัวเดมอนโปรแกรมเองเป็นผู้นำกลุ่มโปรเซส (process group leader) โดยใช้ซิสเต็มคอล setpgrp() ดังนี้

```
#include <sys/types.h>
#include <unistd.h>
pid_t setpgrp(void);
```

และการทำให้โปรเซสไม่ต้องการเทอร์มินัลควบคุม โดยทำการฟอร์คอีกครั้งหนึ่ง

1.4. ทำการปิดตัวบอกไฟล์ (file descriptor) ทั้งหมดที่สามารถเปิดใช้ได้ เพราะว่าอาจจะมีตัวบอกไฟล์ที่ เดมอนโปรแกรม ได้รับมาจากโปรเซสแม่ก่อนทำการฟอร์ค โดยใช้ซิสเต็มคอล close() ดังนี้

```
#include <unistd.h>
int close(filedes);
```

โดยที่ filedes เป็นตัวบอกไฟล์ของระบบ

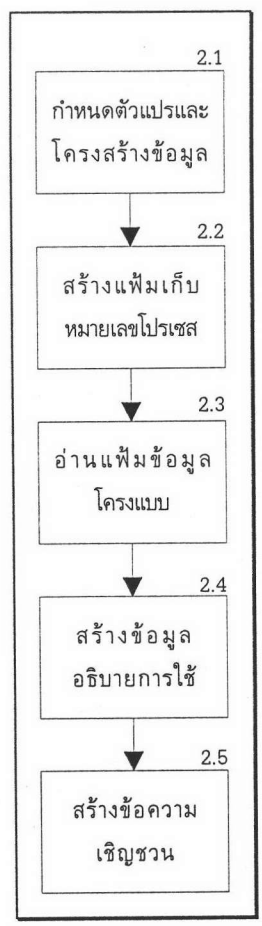
1.5. ทำการย้ายไดเร็กทอรี่ปัจจุบัน ไปอยู่ที่ไดเร็กทอรี่ที่ต้องการ

1.6. กำหนดค่าเริ่มต้นของสิทธิในการเข้าถึงไฟล์ (file mode creation mask) เสียใหม่

1.7. กำหนดให้เดมอนโปรแกรม คอยรอรับสัญญานการสิ้นสุดการทำงาน ของโปรเซสลูก เพื่อไม่ให้โปรเซสลูกกลายไปเป็น โปรเซสที่ไม่ได้รับความสนใจ (zombie process) และคงค้างอยู่ในระบบ ทำให้สูญเสียทรัพยากรของระบบไป

2. การกำหนดค่าเริ่มต้นต่าง ๆ ให้กับโปรแกรม

เมื่อโปรแกรมได้เปลี่ยนการทำงานของตนเองไปเป็น เดมอนโปรแกรมแล้ว จะต้องมีการกำหนดค่าเริ่มต้นต่าง ๆ ให้กับการทำงานต่อไปของโปรแกรม ที่สำคัญคือ



รูปที่ 4.4 แสดงขั้นตอนการกำหนดค่าเริ่มต้นต่าง ๆ แก่โปรแกรม

2.1. ทำการสร้างตัวแปร และโครงสร้างข้อมูลต่าง ๆ ที่ต้องใช้งานในโปรแกรม และอาจจะกำหนดค่าเริ่มต้นให้ด้วย สำหรับโครงสร้างข้อมูลที่สำคัญ คือ โครงสร้างข้อมูลที่ใช้เก็บรายละเอียดของแต่ละโปรแกรมบริการ สำหรับให้ตัวโปรแกรมฟิงเกอร์พลัสเดมอน ใช้เป็นข้อมูลในการติดต่อกับแต่ละโปรแกรมบริการ มีรายละเอียดคือ

```
typedef struct appinfo_tag {
  char  *binaryname;  /* name of application binary file name */
  char  *brhelpname:  /* name of brief help file name */
  char  *helpname;    /* name of help file name*/
  struct appinfo_tag   *nextapp;
} appinfo;
```

ตัวแปรที่ใช้เก็บคำขอข้อมูลที่อ่านเข้ามาได้ คือ

```
char  inq_rcv[BUFSIZ];
char  inq_cook[BUFSIZ];
```

โดย BUFSIZ คือค่าคงที่ของระบบที่หมายถึงขนาดสูงสุดของที่พักข้อมูล (buffer) ของระบบ และ inq_rcv เป็นตัวแปรชนิดตัวอักษรที่มีความจุข้อมูล BUFSIZ ตัวอักษร ใช้เก็บคำขอข้อมูลที่อ่านเข้ามาได้ในเบื้องต้น สำหรับ inq_cook เป็นตัวแปรชนิดตัวอักษรที่มีความจุข้อมูล BUFSIZ ตัวอักษรเช่นกัน ทำหน้าที่เก็บคำขอข้อมูลที่มีมายังระบบฟิงเกอร์พลัสที่ได้ทำการจัดรูปแบบให้พร้อมต่อการวิเคราะห์แล้ว

สำหรับตัวแปรที่ใช้เก็บค่าของ argument1 ที่ผู้ใช้อาจจะระบุมา คือ

```
int   cont;
```

ถ้าค่านี้เป็น 0 หมายถึง ระบบฟิงเกอร์พลัสจะไม่ส่งผลลัพธ์ที่มีความยาวเกินกว่า 1 หน้าจอเทอร์มินัล (ประมาณ 24 บรรทัด) กลับไปยังไคลเอ็นต์ ถ้าผู้ใช้ไม่ระบุค่านี้มา ค่าโดยปริยายของ cont เป็น 0 แต่ถ้าผู้ใช้ระบุ c

มาใน argument1 โปรแกรมจะกำหนดค่าของ cont ให้เป็น 1 มีผลให้ทำการส่งผลลัพธ์กลับไปยังผู้ใช้ไม่ว่าผลลัพธ์นั้นจะมีความยาวเท่าใดก็ตาม

int help;

ถ้าค่านี้เป็น 0 หมายถึง ไม่ต้องการข้อมูลอธิบายการใช้โปรแกรมอย่างย่อ ถ้าผู้ใช้ไม่ระบุค่านี้มา ค่าโดยปริยายของ help เป็น 0 แต่ถ้าผู้ใช้ระบุ h มาใน argument1 โปรแกรมจะกำหนดค่าของ help เป็น 1 มีผลให้โปรแกรมส่งข้อมูลอธิบายการใช้โปรแกรมอย่างย่อไปให้ผู้ใช้

int print;

ถ้าค่านี้เป็น 0 หมายถึง ไม่ต้องการให้ผลลัพธ์ออกทางเครื่องพิมพ์ที่ต่ออยู่โดยตรงกับเทอร์มินัลของผู้ใช้ ค่าโดยปริยายของ print เป็น 0 แต่ถ้าผู้ใช้ระบุ p มาใน argument1 โปรแกรมจะกำหนดค่าของ print เป็น 1 และมีผลให้โปรแกรมส่งผลลัพธ์ที่ได้ออกสู่เครื่องพิมพ์ที่ต่ออยู่โดยตรงกับเทอร์มินัลของผู้ใช้ด้วย

2.2. เก็บข้อมูลหมายเลขกระบวนการของโปรแกรมฟิงเกอร์พลัสเดมอน ไว้ในแฟ้มข้อมูลชื่อ finger+d.pid เพื่อให้สามารถอ้างอิงถึงโปรเซสของฟิงเกอร์พลัสเดมอนได้อย่างง่ายดาย

2.3. อ่านแฟ้มข้อมูลโครงแบบ (configuration file) จากแฟ้ม finger+d.conf ซึ่งเป็นแฟ้มที่เก็บรายละเอียดของโปรแกรมบริการต่าง ๆ ที่มีอยู่ในระบบฟิงเกอร์พลัส โดยทำการอ่านเข้ามาเก็บไว้ในโครงสร้างข้อมูลแบบรายการโยง (linked list) ของตัวแปรชนิด appinfo โดยที่แต่ละรายการจะเก็บข้อมูลของ 1 โปรแกรมบริการ

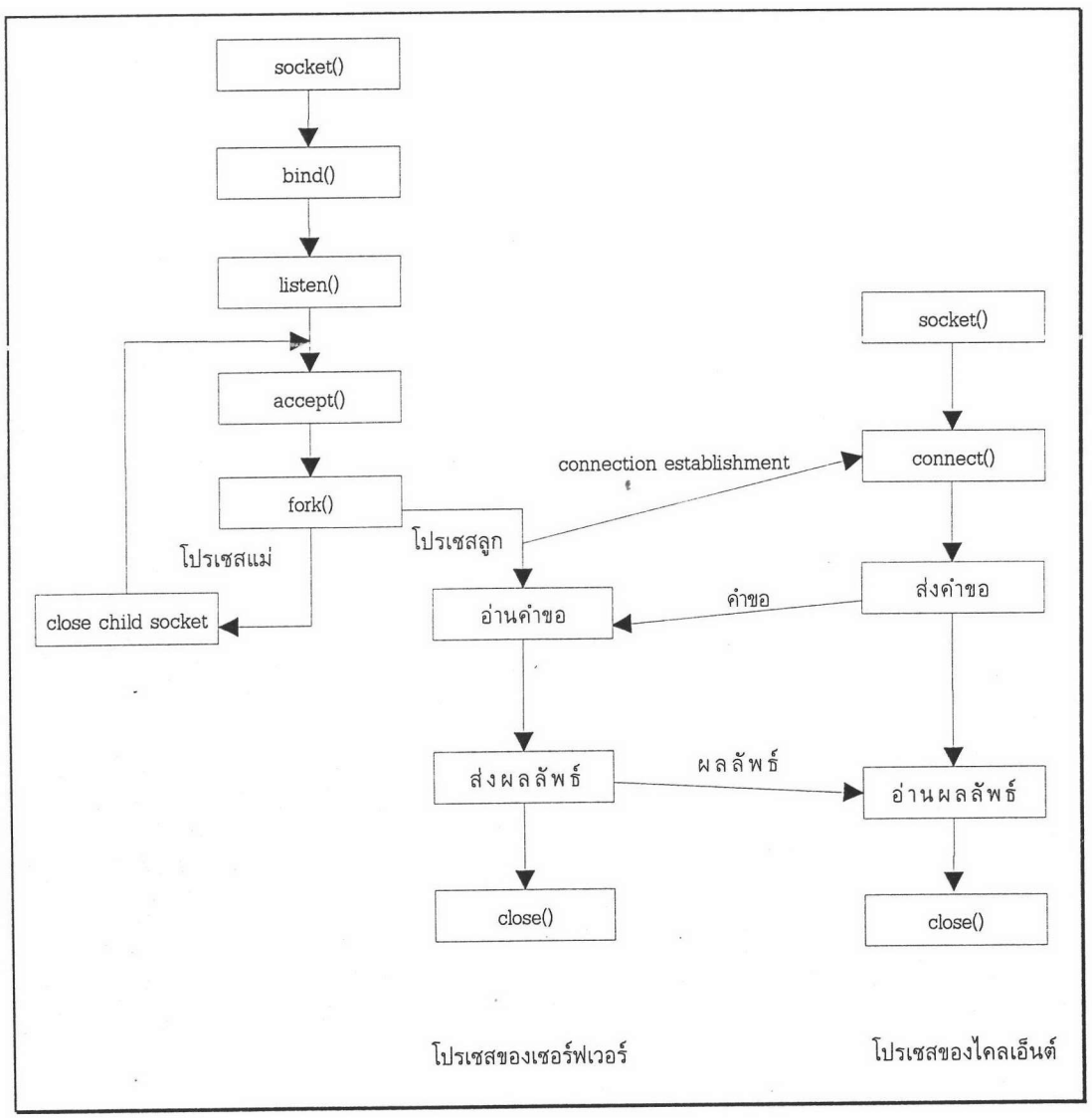
2.4. สร้างแฟ้มข้อมูลคำอธิบายโปรแกรมอย่างย่อ (brief help file) และแฟ้มข้อมูลคำอธิบายโปรแกรม (help file) โดยการนำเอาแฟ้มข้อมูลต้นแบบของคำอธิบายโปรแกรมอย่างย่อ และแฟ้มข้อมูลต้นแบบของคำอธิบายโปรแกรม มาผนวกรวมกับแฟ้มข้อมูลคำอธิบาย

โปรแกรมของแต่ละโปรแกรมบริการ เพื่อส่งให้ผู้ใช้ที่ขอคำอธิบายโปรแกรมมา ได้ทราบว่าชุดโปรแกรมฟิงเกอร์พลัสนี้มีโปรแกรมบริการใดให้เรียกใช้ได้บ้าง

2.5. สร้างข้อความเชิญชวน (invite message) โดยที่ข้อความนี้จะส่งให้กับผู้ใช้ที่ขอใช้บริการของระบบฟิงเกอร์แบบเดิมเข้ามา เพื่อให้ผู้ใช้ทราบว่าตนเองได้ติดต่ออยู่กับระบบฟิงเกอร์พลัส และบอกถึงวิธีการเรียกข้อมูลคำอธิบายโปรแกรมอย่างย่อของระบบฟิงเกอร์พลัส เป็นการแนะนำเบื้องต้นแก่ผู้ใช้ถึงวิธีการใช้งานระบบฟิงเกอร์พลัส

3. สร้างซอกเกตเพื่อรอรับการติดต่อจากไคลเอ็นต์

ในการพัฒนาโปรแกรมฟิงเกอร์พลัสเดมอน โดยใช้ภาษาซี ได้ใช้ เอพีไอ (application programming interface, API) แบบ Berkeley socket interface ซึ่งเป็น เอพีไอสำหรับการสื่อสารเครือข่าย (communication API) ที่ใช้กันมากในระบบยูนิกซ์ โดยรูปที่ 4.5 เป็นการสรุปถึงขั้นตอนในการใช้ชุดคำสั่งซอกเกตในการสื่อสารแบบ connection oriented (Stevens,1991)



รูปที่ 4.5 การใช้ชุดคำสั่งซอกเกตสำหรับการสื่อสารแบบ connection oriented

ต่อไปจะได้อธิบายถึงการใช้ชุดคำสั่งซอกเกตพอสังเขป

3.1. ซอกเกตแอดเดรส (socket address)

ซิสเต็มคอลในระบบเครือข่ายทั่วไป จะต้องการตัวชี้ไปยังโครงสร้างของซอกเกตแอดเดรส เพื่อเป็นอาร์กิวเมนต์ โดยโครงสร้างนี้ได้ถูกกำหนดไว้ใน <sys/socket.h> ดังนี้

```
struct sockaddr {
  u_short     sa_family;      /* address family: AF_xxx value */
  char        sa_data[14]     /* up to 14 bytes of protocol-specific
                                 address */
};
```

สำหรับเนื้อที่ 14 ไบท์ ที่เป็นเนื้อที่สำหรับ protocol-specific address นั้น จะถูกแปลตามชนิดของแอดเดรส ซึ่งสำหรับอินเตอร์เนทแฟมิลี่ (internet family) ได้กำหนดโครงสร้างไว้ใน <netinet/in.h> ดังนี้

```
struct in_addr  {
  u_long      s_addr;         /* 32-bit netid/hostid */
                              /* network byte ordered */
};
struct sockaddr_in  {
  short       sin_family;     /* AF_INET */
  u_short     sin_port; /* 16-bit port number */
                        /* network byte ordered */
  struct in_addr    sin_addr; /* 32-bit netid/hostid */
                              /* network byte ordered */
  char        sin_zero[8];    /* unused */
};
```

ใน <sys/types.h> ได้กำหนดชื่อเพื่อใช้แทนประเภทข้อมูลจำนวนเต็มแบบไม่มีเครื่องหมายไว้แตกต่างกันในระบบ BSD และ System V ไว้ดังนี้

ชนิดข้อมูลในภาษาซี	ระบบ BSD	ระบบ SystemV
unsigned char	u_char	unchar
unsigned short	u_short	ushort
unsigned int	u_int	uint
unsigned long	u_long	ulong

ตารางที่ 4.1 การกำหนดประเภทข้อมูลจำนวนเต็มแบบไม่มีเครื่องหมาย

3.2. ซอกเกตซิสเต็มคอลพื้นฐาน

ต่อไปจะได้กล่าวถึงซิสเต็มคอลพื้นฐานที่จำเป็นต้องใช้ในการพัฒนาโปรแกรมบนระบบเครือข่าย

3.2.1. ซิสเต็มคอล socket

ซิสเต็มคอล socket เป็นสิ่งแรกที่โปรเซสที่ต้องการทำงานอินพุทและเอาพุท ผ่านระบบเครือข่ายจะต้องเรียกใช้ มีรูปแบบการใช้งานดังนี้

```
#include  <sys/types.h>
#include  <sys/socket.h>
int socket (int family, int type, int protocol)
```

ในที่นี้ โปรโตคอลของการสื่อสารข้อมูลที่ใช้เป็นแบบ TCP ค่าของ family จึงเป็น AF_INET และ ค่าของ type เป็น SOCK_STREAM สำหรับ internet protocol แบบ stream socket สำหรับค่าของ protocol โดยปกติแล้วค่าเป็น 0

ซิสเต็มคอล socket นี้จะส่งค่าจำนวนเต็มที่คล้ายกับตัวบอกแฟ้ม กลับไปยังผู้เรียกใช้ เรียกค่านี้ว่า ตัวบอกซอกเกต (socket descriptor) หรือ sockfd เพื่อใช้ในการอ้างถึง socket นี้ต่อไป

3.2.2. ซิสเต็มคอล bind

ซิสเต็มคอล bind ใช้กำหนดชื่อให้กับซอกเกตที่ยังไม่มีชื่อ มีรูปแบบการใช้ดังต่อไปนี้

```
#include <sys/types.h>
#include <sys/socket.h>
int bind (int sockfd, struct sockaddr *myaddr, int addrlen);
```

อาร์กิวเมนต์ตัวที่ 2 คือ ตัวชี้ที่ชี้ไปยัง protocol-specific address และอาร์กิวเมนต์ตัวที่ 3 คือขนาดของโครงสร้างข้อมูลแอดเดรส ซิสเต็มคอล bind สามารถใช้งานได้ใน 3 ลักษณะคือ

1. เป็นการที่ เซอร์ฟเวอร์ บอกกับระบบถึงแอดเดรสของตนเอง เพื่อที่ระบบจะได้ส่งต่อข่าวสารที่ได้รับมาให้กับเซอร์ฟเวอร์ได้ สำหรับโปรเซสที่เป็นเซอร์ฟเวอร์จะต้องเรียกใช้ซิสเต็มคอล bind นี้ก่อน จึงจะสามารถรอรับคำขอจากไคลเอ็นต์ได้
2. ใช้ในกรณีที่เป็นไคลเอ็นต์ ระบุถึงแอดเดรสของตนเองต่อระบบ
3. ในกรณีของไคลเอนต์แบบ คอนเนคชั่นเลส ใช้เมื่อต้องการให้แน่ใจว่าระบบได้กำหนดแอดเดรสที่ไม่ซ้ำกันกับโปรเซสอื่นให้ เพื่อที่จะได้รับการติดต่อจากเซอร์ฟเวอร์ได้อย่างถูกต้อง

ในการพัฒนาระบบฟิงเกอร์พลัสนี้ ได้ใช้ซิสเต็มคอล bind ในลักษณะที่ 1 เท่านั้น

3.2.3. ซิสเต็มคอล connect

ซิสเต็มคอล connect นี้ ไคลเอ็นต์เรียกใช้หลังจากเรียกใช้ซิสเต็มคอล socket แล้ว เพื่อเริ่มการติดต่อกับเซอร์ฟเวอร์ มีรูปแบบการใช้งานดังต่อไปนี้

```
#include <sys/types.h>
#include <sys/socket.h>
int connect (int sockfd, struct sockaddr *servaddr, int
        addrlen);
```

โดยที่ sockfd คือ ค่าของตัวบอกซอกเกตที่ได้รับจากการเรียกใช้ซิสเต็มคอล socket และอาร์กิวเมนต์ตัวที่ 2 และ 3 คือตัวชี้ไปยังโครงสร้างของซอกเกตแอดเดรส และขนาดของโครงสร้างดังกล่าวตามลำดับ สำหรับการพัฒนาโปรแกรมในที่นี้ใช้โปรโตคอลแบบทีซีพี ถือเป็นแบบ connection oriented ซึ่งการเรียกใช้ซิสเต็มคอล connect นี้จะเป็นการสร้างเส้นทางการสื่อสารเชื่อมต่อระหว่างระบบท้องถิ่น (local system) และระบบอื่น (foreign system) ที่อาจอยู่ห่างไกล และการที่ไคลเอ็นต์เรียกใช้ซิสเต็มคอล connect นี้ ไคลเอ็นต์ไม่ต้องทำการ bind กับ local address ก่อน

3.2.4. ซิสเต็มคอล listen

เซอร์ฟเวอร์แบบ connection oriented จะเรียกใช้ซิสเต็มคอล listen เพื่อแสดงว่าพร้อมที่จะรับการติดต่อแล้ว มีรูปแบบการใช้งานดังต่อไปนี้

```
int listen (int sockfd, int backlog)
```

โดยปกติแล้วซิสเต็มคอล listen นี้จะถูกเรียกใช้หลังจาก ซิสเต็มคอล socket และ bind และก่อนที่จะเรียกใช้ซิสเต็มคอล accept ทันที โดยที่อาร์กิวเมนต์ backlog หมายถึง จำนวนของคำขอติดต่อของไคลเอ็นต์ที่สามารถให้รออยู่ในขณะที่เซอร์ฟเวอร์กำลังทำงานในซิสเต็มคอล accept อยู่ โดยปกติแล้วค่านี้จะถูกกำหนดให้เป็น 5 ซึ่งเป็นค่าสูงสุดที่ยอมให้เป็นได้

3.2.5. ซิสเต็มคอล accept

หลังจากที่เซอร์ฟเวอร์แบบ connection oriented ได้เรียกใช้ซิสเต็มคอล listen แล้ว จะต้องเรียกใช้ซิสเต็มคอล accept นี้ก่อน ไคลเอ็นต์จึงจะสามารถติดต่อสื่อสารได้กับเซอร์ฟเวอร์ โดยมีรูปแบบการใช้ดังนี้

```
#include <sys/types.h>
#include <sys/socket.h>
int accept (int sockfd, struct sockaddr *peer, int *addrlen);
```

ซิสเต็มคอล accept นี้จะรับการติดต่อเพื่อสร้างเส้นทางการเชื่อมต่อกับไคลเอนต์ โดยการสร้างซอกเกตที่มีคุณสมบัติเหมือนกันทุกประการกับ sockfd ขึ้นมา และส่งค่าจำนวนเต็มของตัวบอกซอกเกตกลับไปยังผู้เรียกใช้ และถ้ายังไม่มีการขอติดต่อมาจากไคลเอนต์ ซิสเต็มคอล accept จะต้องหยุดรอจนกว่าจะมีการขอติดต่อเข้ามา อาร์กิวเมนต์ peer จะใช้ในการส่งที่อยู่ของไคลเอนต์ที่ติดต่อด้วย สำหรับอาร์กิวเมนต์ addrlen จะใช้เก็บขนาดของที่อยู่ที่เก็บอยู่ในอาร์กิวเมนต์ peer ซึ่งอาร์กิวเมนต์ทั้ง 2 นี้จะถูกส่งค่ากลับไปให้ผู้เรียกด้วย

สำหรับเซอร์ฟเวอร์ที่เป็นแบบ คอนเคอเร็นท์เซอร์ฟเวอร์ เมื่อมีการขอติดต่อเข้ามาจากไคลเอ็นต์ และซิสเต็มคอล accept ส่งค่าของตัวบอกซอกเกตที่ได้คืนมาแล้ว โปรเซสจะทำการฟอร์คตัวเอง แล้วให้โปรเซสลูกทำการให้บริการแก่ไคลเอนต์ต่อไป สำหรับโปรเซสแม่ก็จะกลับไปทำงานในซิสเต็มคอล accept เพื่อรอรับการติดต่อที่จะมีมาอีกต่อไป

3.2.6. ซิสเต็มคอล close

ซิสเต็มคอล close ใช้ในการปิดการใช้งานของซอกเกต มีรูปแบบการใช้งานดังนี้

int close (int fd);

สำหรับในโปรโตคอลแบบที่ซีพี ที่ต้องมีการส่งข้อมูลแบบเชื่อถือได้ (reliable delivery) ซิสเต็มคอล close จะกลับคืนสู่ระบบหลังจากที่ถูกเรียกใช้ทันที แต่ถ้ามีข้อมูลที่ยังรอการส่ง หรือรอการตอบรับอยู่ ระบบจะต้องจัดการข้อมูลดังกล่าวให้เรียบร้อยหลังจากที่กลับคืนจากซิสเต็มคอล close แล้ว

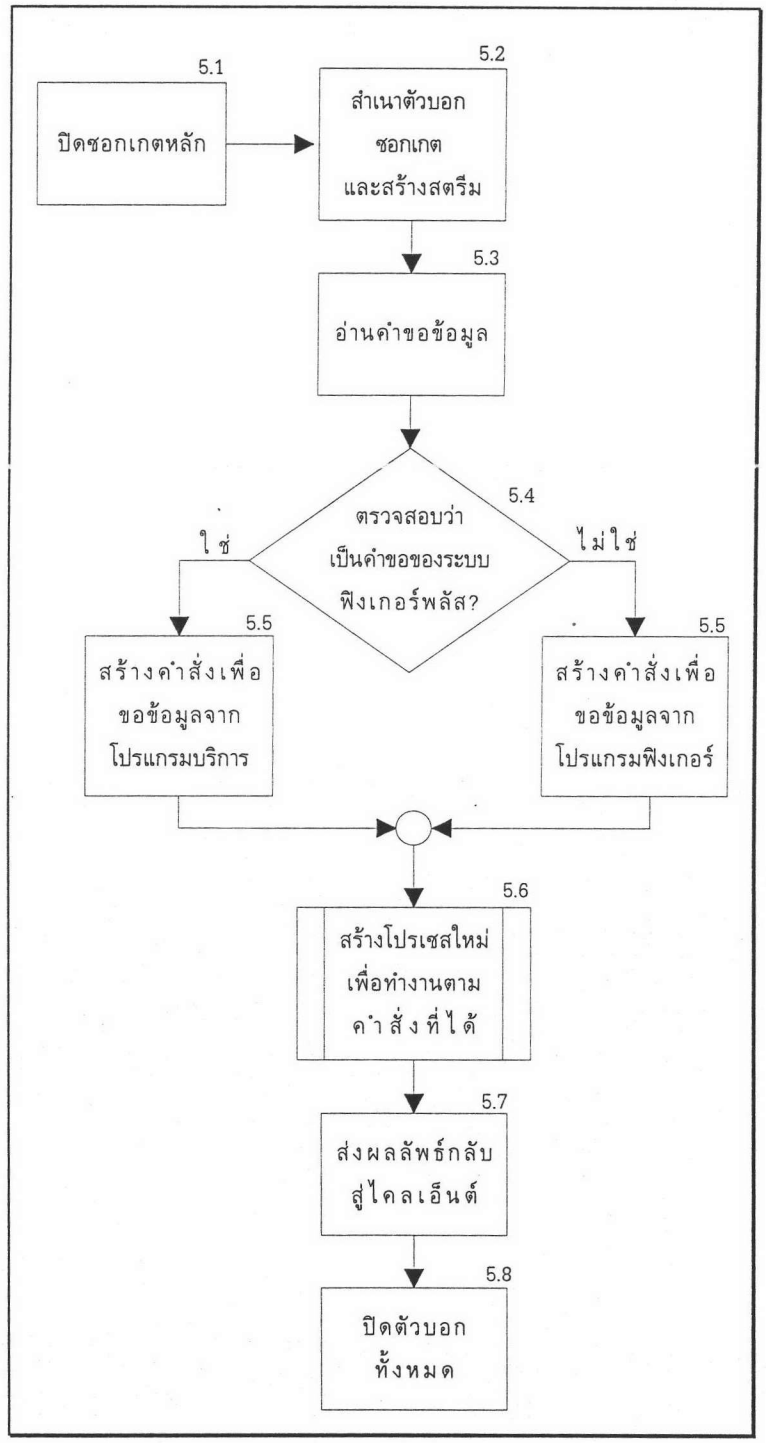
4. สร้างโปรเซสใหม่เพื่อให้บริการกับไคลเอ็นต์ที่ติดต่อเข้ามา

หลังจากที่ได้ทำการสร้างซอกเกตแล้ว ก่อนที่จะเรียกใช้ซิสเต็มคอล accept เพื่อรอรับการขอติดต่อจากไคลเอ็นต์ โปรเซสของเซอร์ฟเวอร์จะเปลี่ยนชื่อเจ้าของโปรเซสจากเดิมซึ่งเป็นผู้จัดการระบบ (root) ไปเป็นชื่อของผู้ใช้อื่น ที่มีสิทธิในระบบรองลงมาจากผู้จัดการระบบ ทั้งนี้เพื่อความปลอดภัย (security) ของระบบ แล้วโปรเซสก็จะเรียกใช้ซิสเต็มคอล accept

เมื่อมีการขอติดต่อมาจากไคลเอ็นต์ และซิสเต็มคอล accept ได้ส่งค่าของตัวบอกซอกเกตใหม่คืนมาให้เรียบร้อยแล้ว โปรเซสของเซอร์ฟเวอร์จะทำการเรียกใช้ซิสเต็มคอล fork เพื่อให้ได้โปรเซสใหม่ หรือโปรเซสลูก และให้โปรเซสลูกจัดการให้บริการแก่ไคลเอ็นต์ต่อไป สำหรับโปรเซสของเซอร์ฟเวอร์ หรือโปรเซสแม่จะทำการปิดซอกเกตใหม่ และวนกลับไปทำซิสเต็มคอล accept อีก เพื่อรอรับการติดต่อต่อไป ตามรูปที่ 4.5

5. โปรเซสลูกทำการให้บริการตามคำขอของไคลเอ็นต์

เมื่อโปรเซสลูกเริ่มต้นทำงาน ในการให้บริการแก่คำขอของไคลเอ็นต์ โดยมีขั้นตอนการทำงานดังแสดงในรูปที่ 4.6



รูปที่ 4.6 แสดงขั้นตอนการให้บริการตามคำขอแก่ไคลเอ็นต์

5.1. ทำการปิดซอกเกตหลัก (master socket) โดยโปรเซสลูกได้รับการถ่ายทอดค่าตัวบอกซอกเกตของซอกเกตหลักมาจากโปรเซสแม่ หลังจากการเรียกใช้ซิสเต็มคอล fork() แล้ว ซึ่งซอกเกตหลักนี้ โปรเซสแม่ต้องใช้ในการรอรับการติดต่อที่จะมีเข้ามาต่อไป ดังนั้นโปรเซสลูกจึงต้องทำการปิดซอกเกตหลักนี้

5.2. สำเนาตัวบอกซอกเกต และสร้างสตรีม ในขั้นตอนนี้เป็นการสำเนาตัวบอกซอกเกตออกเป็น ตัวบอกไฟล์ 2 ตัวเพื่อใช้ในการรับ และส่งข้อมูลแยกจากกัน โดยใช้ซิสเต็มคอล dup() ดังนี้

```
#include  <unistd.h>
int dup(int fildes);
```

โดยที่ filedes เป็นตัวบอกไฟล์ต้นฉบับของการสำเนา ในที่นี้หมายถึงค่าตัวบอกซอกเกตที่ได้จากการเรียกใช้ซิสเต็มคอล accept() และซิสเต็มคอล dup() นี้จะให้ค่าของตัวบอกไฟล์ใหม่ซึ่งมีตัวชี้ไฟล์ (file pointer) เดียวกัน

เมื่อได้ค่าของตัวบอกไฟล์สำหรับรับข้อมูลเข้า และส่งข้อมูลออกแล้ว ทำการเปิดสตรีม (stream) จากตัวบอกไฟล์ โดยใช้ฟังก์ชั่นมาตรฐานทางด้านอินพุทเอาท์พุท คือ fdopen() ดังนี้

```
#include <stdio.h>
FILE *fdopen(int fildes,char *type)
```

โดยที่ filedes คือตัวบอกไฟล์ที่ต้องการให้เชื่อมโยงเข้ากับสตรีมที่สร้างขึ้น และ type เป็นตัวชี้ไปยังข้อมูลประเภทตัวอักษร ที่บอกถึงประเภทของสตรีมที่ต้องการ ซึ่งสำหรับสตรีมที่ต้องการให้ทำหน้าที่รับข้อมูลเข้า จะระบุประเภทเป็น “r” และสตรีมที่ต้องการให้ทำหน้าที่ส่งผลลัพธ์ออก จะระบุประเภทเป็น “w” ฟังก์ชั่น fdopen() นี้จะให้ค่าเป็นตัวชี้ไปยังสตรีมที่สร้างขึ้น

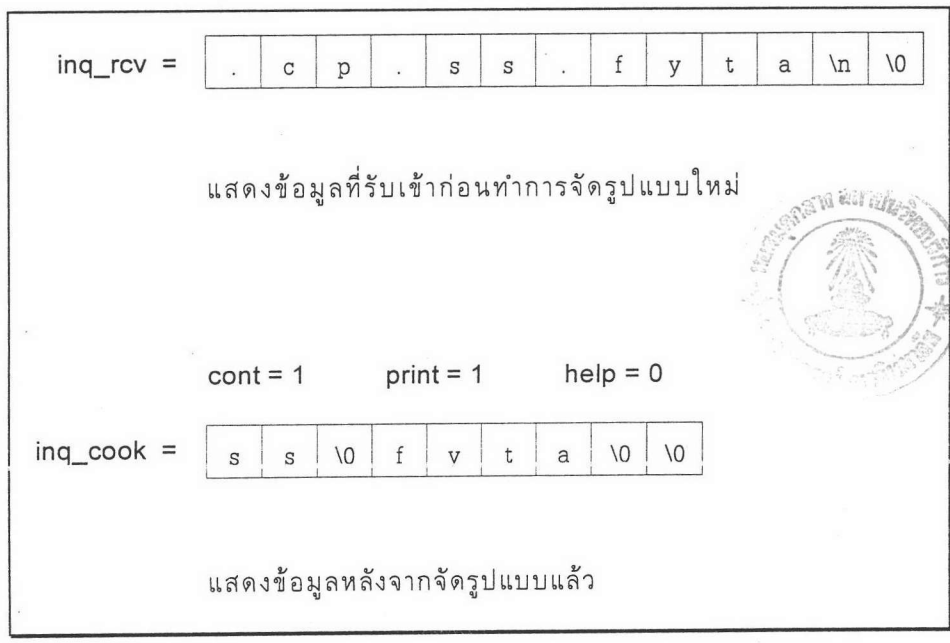
5.3. อ่านคำขอข้อมูล ในขั้นตอนนี้โปรเซสทำการอ่านคำขอข้อมูลผ่านสตรีมที่สร้างขึ้น เข้ามาจำนวน 1 บรรทัด เก็บไว้ในตัวแปร inq_rcv[] โดยใช้ฟังก์ชั่นมาตรฐานอินพุทเอาท์พุท คือ fgets() ดังนี้

```
#include <stdio.h>
char *fgets (char *s, int n, FILE *stream);
```

ฟังก์ชั่น fgets() จะทำการอ่านข้อมูลแบบตัวอักษร จำนวน n-1 ตัวอักษร จากสตรีมที่ถูกชี้โดยตัวชี้ stream หรือ อ่านข้อมูลเข้าจนกว่าจะพบอักขระปัดแคร่ (return character, '\n') เข้ามาเก็บไว้ในตัวแปรแถวลำดับชนิดอักษร (array of character) ที่ชี้โดยตัวชี้ s และทำการปิดท้ายอักขระตัวสุดท้ายด้วยอักขระว่าง (null character, '\0') ฟังก์ชั่น fgets() ให้ค่าเป็น ตัวชี้ s

5.4. ตรวจสอบว่าเป็นคำขอของระบบฟิงเกอร์พลัสหรือไม่ โดยในขั้นตอนนี้เป็นการตรวจสอบคำขอที่ได้รับเข้ามา โดยอาศัยหลักของรูปแบบการสอบถามที่ได้กำหนดขึ้นคือ ถ้ามีอักขระจุด (.) นำหน้าคำขอแล้วจะถือว่าเป็นคำขอที่มีมายังระบบฟิงเกอร์พลัส ถ้าไม่มีอักขระจุด (.) นำหน้าก็ถือว่าเป็นคำขอของระบบฟิงเกอร์แบบเดิม

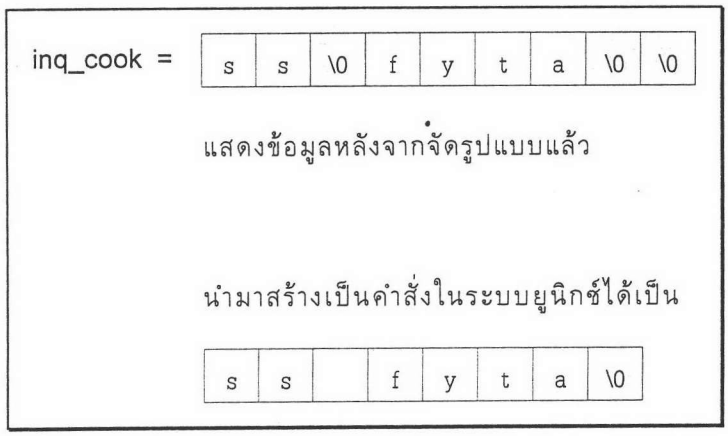
ถ้าหากว่าเป็นคำขอข้อมูลที่มีมายังระบบฟิงเกอร์พลัส จะทำการวิเคราะห์ argument1 ว่าไคลเอ็นต์ระบุคำสั่งใดมาบ้าง โดยเก็บค่าที่ได้ไว้ในตัวแปรเก็บค่าของ อาร์กิวเมนต์1 แล้วทำการแบ่งคำขอข้อมูลที่เหลือออกเป็น สายอักขระที่ปิดท้ายด้วยอักขระว่าง (null terminated string) ที่เรียงต่อกัน และเก็บอยู่ในตัวแปร inq_cook (ดังแสดงในรูปที่ 4.7) เพื่อให้การสร้างคำสั่งเรียกโปรแกรมบริการเป็นไปได้ง่ายขึ้น



รูปที่ 4.7 แสดงรูปแบบการเก็บคำขอข้อมูลก่อน และหลังการจัดรูปแบบ

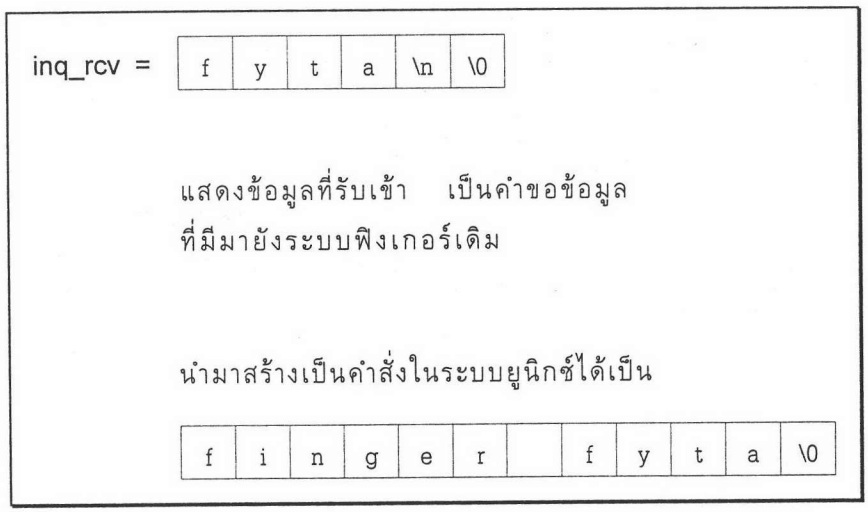
5.5. ขั้นตอนการสร้างคำสั่งเพื่อขอข้อมูลจากโปรแกรมบริการ หรือจากโปรแกรมฟิงเกอร์ ในขั้นตอนนี้ ถ้าหากว่าเป็นคำขอข้อมูลที่มีมายังระบบฟิงเกอร์พลัส โปรแกรมจะทำการตรวจสอบชื่อโปรแกรมบริการที่ระบุมา ว่ามีอยู่ในรายการให้บริการของระบบหรือไม่ ถ้าหากว่าไม่มีจะทำการส่งข้อมูลอธิบายการใช้โปรแกรมอย่างย่อ และข้อมูลโปรแกรมบริการที่มีให้บริการกลับไปยังไคลเอ็นต์ แต่ถ้าหากว่ามีโปรแกรมบริการที่ระบุ จะทำการสร้างคำสั่งที่เหมาะสมในระบบยูนิกซ์ โดยเป็นคำสั่งที่ใช้เพื่อค้นหาข้อมูลจากโปรแกรมบริการนั้น ๆ ตามที่ไคลเอ็นต์ระบุมาในคำขอ

วิธีการสร้างคำสั่ง โดยการนำเอาชื่อของโปรแกรมบริการที่ระบุมา นำมาเรียงต่อด้วยส่วนของ อาร์กิวเมนต์2 ที่มี (อาร์กิวเมนต์2 เป็นส่วนที่ผู้ขอข้อมูลต้องการส่งให้กับโปรแกรมบริการ) ดังรูปที่ 4.8



รูปที่ 4.8 แสดงการสร้างคำสั่งในระบบยูนิกซ์เพื่อเรียกใช้โปรแกรมบริการ

สำหรับคำขอที่มีมายังระบบฟิงเกอร์แบบเดิมนั้น การสร้างคำสั่งเพื่อเรียกให้โปรแกรมฟิงเกอร์ทำการค้นหาข้อมูลผู้ใช้ให้ ทำได้ดังรูปที่ 4.9



รูปที่ 4.9 แสดงการสร้างคำสั่งในระบบยูนิกซ์เพื่อเรียกใช้โปรแกรมฟิงเกอร์

5.6. เมื่อสร้างคำสั่งที่จะเรียกให้โปรแกรมบริการ หรือโปรแกรมฟิงเกอร์ค้นหาข้อมูลให้เรียบร้อยแล้ว โปรแกรมจะทำการสร้างโปรเซสใหม่เพื่อทำงานตามคำสั่งดังกล่าว โดยใช้ฟังก์ชั่นมาตรฐานอินพุทเอาท์พุทของระบบ คือ popen() ดังนี้

```
#include  <stdio.h>
FILE *popen (const char *command, const char *type);
```

ฟังก์ชั่น popen() ทำหน้าที่สร้างท่อการนำข้อมูล (pipe) ระหว่างโปรเซสที่ทำการเรียก กับโปรเซสที่สร้างขึ้นใหม่จากการปฏิบัติงาน (execute) ตามคำสั่งที่ชี้โดย command โดยที่ command เป็นตัวชี้ไปยังค่าคงที่แบบอักษร หมายถึง คำสั่งที่ต้องการจะให้ปฏิบัติงาน (execute) และ type เป็นตัวชี้ไปยังค่าคงที่แบบอักษร ที่บอกถึงประเภทของสตรีมที่ต้องการ ซึ่งสำหรับสตรีมที่ต้องการให้ทำหน้าที่รับข้อมูลเข้า ต้องระบุประเภทเป็น “r” และสตรีมที่ต้องการให้ทำหน้าที่ส่งผลลัพธ์ออก ให้ระบุประเภทเป็น “w” ในที่นี้จะเรียกใช้ฟังก์ชั่น popen() โดยระบุ type เป็น “r” เพื่ออ่านข้อมูลที่ได้เข้ามา ฟังก์ชั่น popen() นี้ให้ค่าเป็นตัวชี้ไปยังสตรีมที่สร้างขึ้น

5.7. เมื่อโปรแกรมปฏิบัติการคำสั่งที่ได้แล้ว จะทำการอ่านผลลัพธ์ที่ได้จากโปรเซสที่สร้างขึ้น โดยอ่านจากสตรีมที่ได้รับมา ทำการอ่านผลลัพธ์ที่ได้เข้ามาจนหมด โดยเรียกใช้ฟังก์ชั่นมาตรฐานอินพุทเอาท์พุท คือ fread() ดังนี้

```
#include <stdio.h>
size_t fread (void *ptr, size_t size, size_t nitems,
                FILE *stream);
```

ฟังก์ชั่น fread() จะทำการอ่านชุดข้อมูลจำนวน nitems ชุด (items) มีขนาดชุดละ size ไบต์ (bytes) จากสตรีมที่ชี้โดยตัวชี้ stream เข้ามาเก็บไว้ในตัวแปรแถวลำดับซึ่งถูกชี้อยู่โดยตัวชี้ ptr ฟังก์ชั่น fread() ให้ค่าเป็นจำนวนชุดของข้อมูลที่อ่านได้

เมื่อโปรแกรมอ่านผลลัพธ์เข้ามาจนหมดแล้ว จะทำการตรวจสอบขนาดของข้อมูลผลลัพธ์ที่ได้ว่ามีขนาดจำนวนบรรทัดเป็นจำนวนเท่าใด พร้อมทั้งตรวจสอบค่าของตัวแปร cont ด้วย ถ้าหากว่าข้อมูลมีจำนวนบรรทัดเกิน 1 หน้าจอของผู้ใช้ (24 บรรทัด) หรือตัวแปร cont มีค่าเป็น 0 จะทำการส่งข้อความเตือนให้ผู้ใช้ทราบถึงจำนวนหน้าของข้อมูลผลลัพธ์ และแจ้งให้ส่งคำขอข้อมูลมาใหม่ พร้อมทั้งเตือนให้ผู้ใช้ระบุ “c” หรือ “p” เป็นอาร์กิวเมนต์1 ด้วย

แต่ถ้าหากว่า ข้อมูลผลลัพธ์มีจำนวนบรรทัดไม่เกิน 1 หน้าจอของผู้ใช้หรือผู้ใช้ระบุอาร์กิวเมนต์1 เป็น "c" หรือ "p" มาในคำขอ จะทำการส่งข้อมูลผลลัพธ์ทั้งหมดกลับไปยังไคลเอ็นต์ โดยเรียกใช้ฟังก์ชั่นมาตรฐานอินพุทเอาท์พุท คือ fputs() ดังนี้

```
#include <stdio.h>
int fputs (const char *ptr, FILE *stream);
```

ฟังก์ชั่น fputs() ทำการเขียนข้อมูลแบบตัวอักษรที่ถูกปิดท้ายอักขระตัวสุดท้ายด้วยอักขระว่าง ซึ่งชี้โดยตัวชี้ ptr ไปยังสตรีมที่ถูกชี้โดยตัวชี้ stream ฟังก์ชั่น fputs() ให้ค่าเป็นจำนวนตัวอักษรที่ได้ทำการเขียนไปยังสตรีม

5.8. ทำการปิดตัวบอกไฟล์ หรือตัวบอกซอกเกตทั้งหมดที่ได้เปิดไว้ โดยใช้ซิสเต็มคอล close() และปิดสตรีมต่าง ๆ โดยใช้ฟังก์ชั่นมาตรฐานอินพุทเอาท์พุท คือ fclose() ดังนี้

```
#include <stdio.h>
int fclose (FILE *stream);
```

ส่วนของโปรแกรมบริการ

1. ลักษณะโดยทั่วไปของโปรแกรมบริการ

โปรแกรมบริการ โดยทั่วไปเป็นโปรแกรมอรรถประโยชน์ในระบบยูนิกซ์ ที่มีอยู่แล้ว หรืออาจเขียนขึ้นมาใหม่ แล้วนำมาเชื่อมต่อให้อยู่ในรายการโปรแกรมบริการของระบบฟิงเกอร์พลัส เนื่องจากข้อตกลง (protocol) ของระบบฟิงเกอร์ ที่ไคลเอ็นต์จะส่งคำขอข้อมูลให้กับเซอร์ฟเวอร์เพียงครั้งเดียว จำนวน 1 บรรทัดเท่านั้น แล้วจะรอรับผลลัพธ์ เพื่อนำกลับมาแสดงให้ผู้ใช้ ดังนั้นโปรแกรมบริการจึงต้องมีคุณสมบัติดังต่อไปนี้

1. เป็นโปรแกรมที่ไม่ทำงานในเชิงโต้ตอบ (interactive) กับผู้ใช้ คือทำงานในลักษณะประมวลผลแบบกลุ่ม (batch processing) ในระหว่างการปฏิบัติงาน จะสามารถปฏิบัติงานจนได้ผลลัพธ์โดยไม่ต้องการรับข้อมูลเข้าใด ๆ จากผู้ใช้อีก
2. รับข้อมูลเข้าโดยผ่านทางบรรทัดคำสั่ง (command line) เท่านั้น
3. เมื่อประมวลผลเสร็จสิ้นแล้ว ต้องส่งผลลัพธ์ออกทางช่องส่งข้อมูลออกมาตรฐาน (standard output) เท่านั้น

นอกจากตัวโปรแกรมบริการเองแล้ว ต้องมีแฟ้มข้อความ (text file) อีก 2 แฟ้ม โดยเป็นแฟ้มเก็บรายละเอียดอย่างย่อของโปรแกรมบริการนั้น ๆ ขนาด 1 บรรทัด จำนวน 1 แฟ้ม และแฟ้มอธิบายรายละเอียดอย่างสมบูรณ์อีกจำนวน 1 แฟ้ม ข้อกำหนดของแต่ละแฟ้มดูได้จากภาคผนวก ก

2. โปรแกรมบริการตัวอย่าง

ในการพัฒนาระบบฟิงเกอร์พลัส ได้ทำการพัฒนาโปรแกรมบริการตัวอย่างขึ้นมาจำนวน 3 โปรแกรม คือ

2.1 help

โปรแกรมบริการ help เป็นโปรแกรมบริการที่จำเป็นจะต้องมีอยู่ในระบบฟิงเกอร์พลัสเสมอ เพราะเป็นส่วนที่ทำหน้าที่ในการให้ข้อมูลอธิบายการใช้คำสั่งของโปรแกรมบริการอื่น ๆ โดยรับข้อมูลเข้าเป็นชื่อของโปรแกรมบริการที่ต้องการทราบรายละเอียดการใช้งาน และทำการอ่านแฟ้มข้อมูลรายละเอียดของโปรแกรมบริการนั้น ๆ ส่งกลับมาให้

2.2 man

โปรแกรมบริการ man ทำหน้าที่ในการแสดง คู่มือการใช้คำสั่ง (manual page) ในระบบยูนิกซ์ โดยรับข้อมูลเข้าเป็นชื่อของคำสั่งที่ต้องการทราบรายละเอียด แล้วทำการเรียกโปรแกรม man ของระบบยูนิกซ์ทำการค้นหาข้อมูลให้อีกขั้นหนึ่ง

2.3 ss

โปรแกรมบริการ ss (substring search user information) เป็นโปรแกรมที่ให้บริการข้อมูลของผู้ใช้ ในทำนองเดียวกันกับโปรแกรมฟิงเกอร์ไคลเอนต์ แต่สามารถให้บริการข้อมูลได้กว้างขวางกว่า โดยครอบคลุมถึงข้อมูลของผู้ใช้ที่ถูกกำหนดอยู่ในแฟ้มข้อมูลสมนาม (aliases file) ของระบบ และสามารถค้นหาข้อมูลผู้ใช้ได้โดยไม่ต้องระบุชื่อผู้ใช้อย่างเต็มคำ ลักษณะการทำงานของโปรแกรม จะรับข้อมูลเข้าเป็น ส่วนของข้อมูลผู้ใช้ (partial string of user information) แล้วทำการค้นหารายชื่อผู้ใช้ที่มีคำดังกล่าวประกอบอยู่ จากแฟ้มข้อมูลผู้ใช้ระบบ (/etc/passwd file) แล้วทำการเรียกโปรแกรมฟิงเกอร์ของระบบยูนิกซ์ทำการค้นหาข้อมูลให้อีกขั้นหนึ่ง จากนั้นทำการค้นหารายชื่อผู้ใช้จากแฟ้มข้อมูลสมนามของระบบ แล้วนำผลลัพธ์ที่ได้ทั้งหมดส่งกลับมาให้ผู้ขอ

3. ตัวอย่างขั้นตอนการบรรจุโปรแกรมบริการ

การบรรจุโปรแกรมบริการใหม่เข้าสู่ระบบฟิงเกอร์พลัส มีขั้นตอนการดำเนินการเป็นขั้น ๆ โดยตัวอย่างการบรรจุโปรแกรมบริการชื่อ test เป็นดังนี้

3.1 เตรียมแฟ้มโปรแกรมบริการ test โดยการแปลโปรแกรมให้อยู่ในรูปแบบที่สามารถทำงาน (run) ได้ และแฟ้มอธิบายรายละเอียดอีก 2 แฟ้ม คือ help.test และ help.test.br

3.2 ย้ายแฟ้มต่าง ๆ ไปอยู่ในไดเรกทอรีย่อยที่เหมาะสม

3.3 ทำการเพิ่มเติมรายชื่อของโปรแกรมบริการ test ในแฟ้มข้อมูลโครงแบบของฟิงเกอร์พลัส คือ แฟ้ม finger+d.conf

3.4 เริ่มต้นการทำงาน (restart) ของโปรแกรมฟิงเกอร์พลัสเดมอน อีกครั้ง เพื่อให้โปรแกรมรับรู้ถึงการบรรจุเข้ามาใหม่ของโปรแกรมบริการ test

สำหรับรายละเอียดของแฟ้มข้อมูล และไดเรกทอรีย่อยต่าง ๆ ได้แสดงอยู่ในภาคผนวก ก

ระบบของการให้ข้อมูลอธิบายการใช้คำสั่ง

ระบบฟิงเกอร์พลัส มีการให้ข้อมูลอธิบายการใช้คำสั่งแก่ผู้ใช้ โดยแบ่งเป็น 2 ชนิดคือ

1. การให้ข้อมูลอธิบายการใช้คำสั่งโดยที่ผู้ใช้ไม่ต้องร้องขอ

ระบบฟิงเกอร์พลัส มีการให้ข้อมูลแนะนำวิธีการเรียกข้อมูลอธิบายการใช้คำสั่งกลับไปให้กับผู้ที่ขอข้อมูลมา โดยที่ผู้ใช้ไม่ต้องทำการร้องขออยู่ 2 ลักษณะคือ

1.1 เมื่อผู้ใช้ทำการขอข้อมูลจากระบบฟิงเกอร์พลัส โดยใช้รูปแบบคำขอของระบบฟิงเกอร์แบบเดิม ระบบฟิงเกอร์พลัสจะส่ง ข้อความเชิญชวน (invite message) ซึ่งเป็นข้อความที่บอกให้ผู้ใช้ได้ทราบว่าตนเองกำลังติดต่ออยู่กับระบบฟิงเกอร์พลัส และสามารถเรียกใช้ระบบฟิงเกอร์พลัสได้อย่างไร โดยข้อความนี้จะส่งกลับมาหลังจากส่งผลลัพธ์แล้วทุกครั้ง

1.2 เมื่อคำขอที่ผู้ใช้ส่งมายังระบบฟิงเกอร์พลัส มีรูปแบบที่ไม่ถูกต้องหรือระบุชื่อของโปรแกรมบริการที่ไม่มีให้บริการ ระบบฟิงเกอร์พลัสจะส่งข้อความอธิบายการใช้

คำสั่งแบบย่อ ซึ่งอธิบายถึงรูปแบบของคำขอที่ถูกต้อง และรายชื่อโปรแกรมบริการที่มีให้บริการกลับมายังผู้ใช้

ซึ่งระบบการให้ข้อมูลอธิบายการใช้คำสั่งแบบผู้ใช้ไม่ต้องทำการร้องขอนี้ เป็นการให้บริการโดยตัวโปรแกรมฟิงเกอร์พลัสเดมอนเอง ไม่ได้เรียกใช้โปรแกรมบริการ help

2. การให้ข้อมูลอธิบายการใช้คำสั่งเมื่อมีการร้องขอ

ระเบียบการให้ข้อมูลอธิบายการใช้คำสั่งแบบนี้ จะแสดงต่อผู้ใช้เมื่อผู้ใช้ทำการร้องขอผ่านมาทางคำขอข้อมูล แบ่งได้ 2 ลักษณะคือ

2.1 โปรแกรมฟิงเกอร์พลัสเดมอน เป็นผู้ให้บริการเอง การให้ข้อมูลในลักษณะนี้เกิดขึ้นเมื่อ ผู้ใช้ส่งคำขอข้อมูลที่ระบุอาร์กิวเมนต์1 เป็น "h" เมื่อโปรแกรมฟิงเกอร์พลัสเดมอนได้รับคำขอนี้ ก็จะดำเนินการส่งข้อมูลอธิบายการใช้โปรแกรมแบบย่อกลับมายังผู้ใช้

2.2 โปรแกรมฟิงเกอร์พลัสเดมอน ทำการเรียกโปรแกรมบริการ help ทำการจัดหาข้อมูลให้ ในกรณีนี้จะเกิดขึ้นเมื่อผู้ใช้ส่งคำขอข้อมูลที่ไม่ได้ระบุอาร์กิวเมนต์1 เป็น "h" มา แต่ได้ระบุเรียกใช้โปรแกรมบริการ help ซึ่งถ้าในคำขอได้ระบุอาร์กิวเมนต์2 เป็นชื่อของโปรแกรมบริการที่มีอยู่ โปรแกรม help ก็จะส่งข้อมูลอธิบายการใช้โปรแกรมบริการดังกล่าวกลับมายังผู้ใช้ แต่ถ้าในคำขอไม่ได้มีการระบุอาร์กิวเมนต์2 โปรแกรม help ก็จะส่งข้อมูลอธิบายการใช้คำสั่งของระบบฟิงเกอร์พลัสแบบสมบูรณ์กลับมายังผู้ใช้

ในบทต่อไปเป็นการกล่าวถึง การทดสอบการทำงานของโปรแกรมฟิงเกอร์พลัสเดมอนภายใต้ระบบปฏิบัติการยูนิกซ์ บนเครื่องคอมพิวเตอร์รุ่นต่าง ๆ