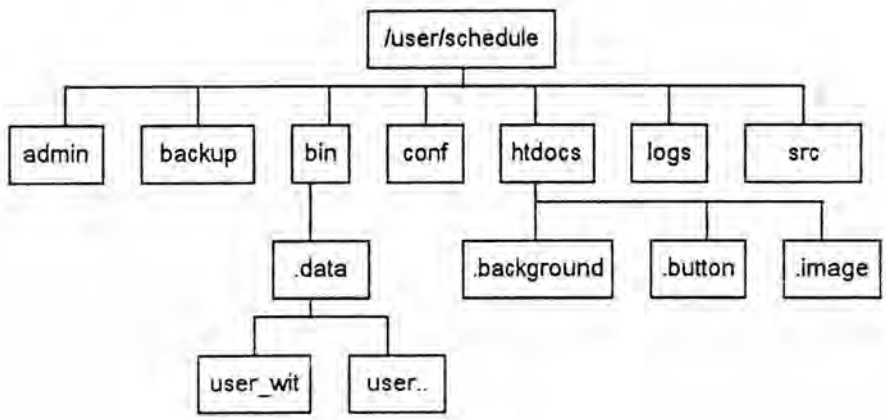
บทที่ 3
โครงสร้างการทำงานในส่วนของเครื่องบริการ

โปรแกรมส่วนเครื่องบริการจะทำงานอยู่บนระบบยูนิกซ์ และใช้โปรแกรม HTTPD ของบริษัท Apache เป็นตัวจัดการ ส่วนตัวโปรแกรมจะถูกพัฒนาขึ้นด้วยภาษา Perl โดยใช้มาตรฐานของ Fast CGI แต่คุณสมบัติการทำงานแบบ Fast CGI นั้นไม่ได้เป็นคุณสมบัติของ Apache HTTPD ต้นฉบับ แต่จะเป็นโมดูล (Module) ที่ถูกแยกเขียนต่างหากโดยบริษัท Open Market ดังนั้นการที่จะใช้งาน Apache HTTPD ร่วมกันกับโมดูล Fast CGI นั้นจำเป็นที่จะต้องนำมาคอมไพล์รวมกัน ส่วน perl interpreter ก็เช่นกัน จำเป็นที่จะต้องใช้ Perl ที่รองรับการเขียนโปรแกรมแบบ Fast CGI ด้วย ซึ่งในที่นี้ได้นำเอาไบนารีโคด (Binary code) ของ Perl เวอร์ชั่น 5.002 ที่รองรับ Fast CGI มาใช้งานด้วย โดยโคดดังกล่าวได้ถูกนำมาจากบริษัท Open Market เช่นกัน

3.1 โครงสร้างไดเรกทอรี

ผู้วิจัยได้กำหนด account บนระบบยูนิกซ์เพื่อให้โปรแกรมเครื่องบริการทำงานโดยเฉพาะ ซึ่ง account นี้มีชื่อว่า schedule โดยมีโฮมไดเรกทอรี (Home directory) คือ /user/schedule โดยจะมีไดเรกทอรีย่อยดังแสดงในรูปที่ 3.1



รูปที่ 3.1 โครงสร้างของไดเรกทอรี

3.1.1 **admin** เป็นไดเรกทอรีที่เก็บโปรแกรมใช้สำหรับการบริหาร CSS ซึ่งโปรแกรมดังกล่าวมีดังต่อไปนี้

- **css_start** เป็นโปรแกรมสำหรับเริ่มต้นการทำงานของ HTTPD และโปรแกรม Fast CGI ที่มีชื่อว่า kernel ซึ่ง kernel นี้จะเป็นโปรแกรมที่รวมการทำงานในส่วนเครื่องบริการทั้งหมดของ CSS ไว้ในแฟ้มเดียว โดยจะทำงานในลักษณะของเดมอนโปรเซสเพื่อรอรับการร้องขอจากเครื่องใช้บริการ
- **css_stop** เป็นโปรแกรมใช้สำหรับหยุดการทำงานของ HTTPD และ kernel
- **css_backup** เป็นโปรแกรมที่ใช้สำหรับการเก็บสำรองข้อมูล (Backup) ตารางเวลาของผู้ใช้ทุกคน โดยโปรแกรมนี้จะไม่ถูกเรียกใช้โดยตรงแต่จะถูกใช้โดยผ่าน crontab ของระบบยูนิกซ์ ซึ่งใช้ตั้งเวลาให้รันโปรแกรมนี้ทุก ๆ วันตอนเที่ยงคืน ทั้งนี้เพื่อป้องกันในกรณีที่ข้อมูลของผู้ใช้เกิดการเสียหายหรือสูญหาย

3.1.2 **backup** เป็นไดเรกทอรีที่เก็บสำรองข้อมูลตารางเวลาของผู้ใช้ทุกคนไว้ โดยข้อมูลถูกเก็บรวบรวมอยู่ในรูปแบบ tar และถูกบีบอัดข้อมูลด้วย gzip อีกทอดโดยชื่อของแฟ้มข้อมูลในการเก็บสำรองแต่ละครั้งจะมีรูปแบบคือ

ปี-เดือน-วัน.tar.gz

เช่น 1997-09-01.tar.gz เป็นต้น

3.1.3 **bin** เป็นไดเรกทอรีที่เก็บโปรแกรมของส่วนเครื่องบริการและข้อมูลตารางเวลาของผู้ใช้ทุกคน ซึ่งแฟ้มข้อมูลที่อยู่ในไดเรกทอรีนี้มีดังต่อไปนี้

- **kernel** เป็นโปรแกรม Fast CGI ที่ซึ่งจะถูกเรียกใช้ขึ้นมาเป็นเดมอนโปรเซสพร้อมกันกับ HTTPD ซึ่งจะเป็นโปรแกรมหลักของฝั่งเครื่องบริการที่รองรับการร้องขอของเครื่องใช้บริการทุกชนิด โครงสร้างการทำงานของโปรแกรม kernel จะถูกกล่าวถึงภายหลัง
- **.data** เป็นไดเรกทอรีที่เก็บข้อมูลตารางเวลาและรหัสผ่านของผู้ใช้ทุกคน โครงสร้างการจัดเก็บข้อมูลจะถูกกล่าวถึงภายหลัง

3.1.4 **conf** เป็นไดเรกทอรีที่เก็บแฟ้มข้อมูลรูปแบบการทำงานของ HTTPD ซึ่งการตั้งค่าพารามิเตอร์ (Parameter) ต่างๆ ของ HTTPD เป็นสิ่งที่มองข้ามไม่ได้ เพราะจะมีผลกระทบต่อประสิทธิภาพการทำงานของโปรแกรม kernel โดยรวม

3.1.5 **htdocs** เป็นไดเรกทอรีที่เก็บแฟ้มข้อมูล HTML ที่บรรจุโปรแกรม Javascript สำหรับการทำงานในฝั่งเครื่องใช้บริการทั้งหมดไว้ ซึ่งนอกจากแฟ้มข้อมูล HTML แล้วยังมีไดเรกทอรีย่อยอื่นที่เกี่ยวข้องดังนี้

- **.background** เป็นไดเรกทอรีสำหรับเก็บแฟ้มรูปภาพที่เป็นพื้นหลัง (Background) สำหรับใช้งานในกรณีที่ผู้ใช้ต้องการเปลี่ยนข้อมูลตารางเวลาของตนเองให้เป็นแฟ้มข้อมูล HTML ในฝั่งเครื่องใช้บริการ
- **.button** เป็นไดเรกทอรีที่เก็บแฟ้มรูปปุ่มฟังก์ชัน (Function) การทำงานต่างๆ สำหรับให้ผู้ใช้กดเพื่อสั่งการ เช่น ปุ่มสำหรับการสร้างการนัดหมายอันใหม่เป็นต้น
- **.image** เป็นไดเรกทอรีที่เก็บรูปภาพอื่น ๆ ที่นอกเหนือจาก 2 ไดเรกทอรีข้างต้น

3.1.6 **logs** เป็นไดเรกทอรีที่เก็บประวัติการทำงานและการถูกใช้งานของ HTTPD และ kernel ว่ามีการถูกเรียกใช้งานจากเครื่องใช้บริการที่มีไอพีแอดเดรส (IP Address) อะไรและเมื่อไร

3.1.7 **src** เป็นไดเรกทอรีที่เก็บโปรแกรมต้นฉบับภาษา c ของโปรแกรม HTTPD ซึ่งในที่นี้จะมีโมดูล Fast CGI (mod_fastcgi.c) รวมอยู่ด้วย ซึ่งแฟ้มข้อมูลที่อยู่ภายใต้ไดเรกทอรีนี้จะใช้สำหรับคอมไพล์เพื่อสร้างไบนารีของ HTTPD ขึ้นมา

3.2 โครงสร้างการจัดเก็บแฟ้มข้อมูล

หากแยกประเภทของแฟ้มข้อมูลตามลักษณะของการจัดเก็บแล้วมีอยู่ด้วยกัน 2 ประเภท ดังต่อไปนี้

- **แฟ้มข้อมูลที่ลงท้ายด้วย .db** เป็นแฟ้มข้อมูลประเภทที่ถูกใช้งานมากที่สุด ซึ่งจะถูกใช้ในการเก็บ password ของผู้ใช้ , ข้อมูลตารางเวลา เป็นต้น และเป็นแฟ้มประเภทที่มีขนาดไม่แน่นอน ส่วนโครงสร้างนั้นจะเป็นลักษณะเรคคอร์ด (Record) ซึ่งแต่ละเรคคอร์ดจะถูกแยกโดยใช้การตัวอักษรขึ้นบรรทัดใหม่ (End of line character) ส่วนฟิลด์ (Field) ในแต่ละเรคคอร์ดจะถูกแยกโดยใช้ตัวอักษร ":" (Colon)
- **แฟ้มข้อมูลที่ลงท้ายด้วย .cfg** เป็นแฟ้มที่ถูกใช้ในการจัดเก็บข้อมูลประเภทรูปแบบการทำงาน ซึ่งจะมีทั้งของโปรแกรมเครื่องบริการและของผู้ใช้แต่ละคน โดยโครงสร้างการจัดเก็บในแต่ละบรรทัดจะมีรูปแบบดังต่อไปนี้

  parameter=value

ตัวอย่างเช่น

Start_View_Time=9

ซึ่งในที่นี้คือพารามิเตอร์ชื่อ Start_View_Time มีค่าเท่ากับ 9 หรือหมายถึงการแสดงผลของตารางเวลาของเครื่องใช้บริการจะเริ่มต้นที่เวลา 9:00 เป็นต้น

แฟ้มข้อมูลที่ถูกจัดเก็บบนเครื่องบริการนั้นมีอยู่ด้วยกัน 2 ประเภทดังต่อไปนี้

3.2.1 **แฟ้มข้อมูลระบบ** เป็นแฟ้มข้อมูลส่วนกลางที่ผู้ใช้ทุกคนใช้งานร่วมกันมีอยู่ด้วยกันดังต่อไปนี้

- account_[a-z].db เป็นแฟ้มข้อมูลที่จัดเก็บ password ของผู้ใช้ทุกคน ซึ่งจะเป็นกลุ่มแฟ้มข้อมูลจำนวน 26 แฟ้ม ตั้งแต่ account_a.db จนถึง account_z.db โดย password ของผู้ใช้แต่ละคนนั้นจะถูกแยกเก็บไปตามแฟ้มทั้ง 26 นี้ โดยขึ้นอยู่กับตัวอักษรขึ้นต้นของ user name เป็นหลัก ตัวอย่างเช่น หากผู้ใช้มี user name ชื่อ john ข้อมูล password ของ john ก็จะถูกเก็บอยู่ในแฟ้ม account_j.db เป็นต้น สาเหตุที่ต้องแยกข้อมูลออกเป็น 26 แฟ้มเนื่องจากผู้วิจัยต้องการให้ขั้นตอนการล็อกอินของผู้ใช้เร็วขึ้น เนื่องข้อมูลที่ต้องตรวจสอบมีน้อยลง ซึ่งแต่ละเรคคอร์ดคือข้อมูล password ของผู้ใช้ 1 คน ซึ่งมีฟิลด์ต่างๆ ดังต่อไปนี้
  - User_Name หมายถึงชื่อที่ใช้สำหรับขั้นตอนของการล็อกอิน
  - Password หมายถึงรหัสผ่านที่ใช้สำหรับขั้นตอนของการล็อกอิน ซึ่งข้อมูลที่ถูกจัดเก็บในฟิลด์นี้จะถูกเข้ารหัส (Encrypt) โดยเรียกใช้ฟังก์ชันการเข้ารหัสของระบบยูนิกซ์ทั้งนี้เพื่อความปลอดภัยของข้อมูลตารางเวลาของผู้ใช้แต่ละคน
  - Create_Date หมายถึงวันเดือนปีที่เรคคอร์ดของผู้ใช้ได้ถูกสร้างขึ้น
  - Create_From หมายถึงไอพีแอดเดรสของเครื่องใช้บริการที่ใช้ในการสร้างเรคคอร์ดนี้ขึ้น
- server.cfg เป็นแฟ้มข้อมูลสำหรับเก็บค่าพารามิเตอร์ในการควบคุมเครื่องบริการ ซึ่งพารามิเตอร์ดังกล่าวมีอยู่ด้วยกันดังต่อไปนี้
  - Enable_New เป็นพารามิเตอร์ที่บ่งบอกว่าอนุญาตให้มีการสร้างเรคคอร์ดผู้ใช้คนใหม่หรือไม่

- **Enable_Login** เป็นพารามิเตอร์ที่บ่งบอกว่าอนุญาตให้ผู้ใช้ล็อกอินเข้ามาหรือไม่

พารามิเตอร์ Enable_New และ Enable_Login จะถูกใช้ในกรณีที่มีการใช้งานโปรแกรม css_stop เพื่อหยุดการทำงานของ CSS ซึ่งช่วงการหยุดการทำงานของ CSS นี้จะเสียเวลาไปชั่วขณะหนึ่ง ดังนั้นในช่วงเวลาดังกล่าวจะไม่อนุญาตให้ผู้ใช้ล็อกอินเข้ามาในระบบหรือสร้าง account ใหม่โดยเด็ดขาด

3.2.2 **<u>แฟ้มข้อมูลผู้ใช้</u>** เป็นกลุ่มของแฟ้มข้อมูลที่ผู้ใช้แต่ละคนมีเป็นของตนเองซึ่งแฟ้มข้อมูลเหล่านี้จะแยกเก็บไว้ในไดเรกทอรีของผู้ใช้แต่ละคน โดยไดเรกทอรีเหล่านั้นจะขึ้นต้นด้วย "user_" และตามด้วยชื่อของผู้ใช้คนนั้นๆ ตัวอย่างเช่น ผู้ใช้ที่ใช้ชื่อ john แฟ้มข้อมูลของเขาทั้งหมดจะถูกเก็บไว้ในไดเรกทอรีชื่อว่า "user_john" เป็นต้น ซึ่งแฟ้มข้อมูลดังกล่าวมีอยู่ด้วยกันดังต่อไปนี้

- <u>appointment.db</u> เป็นแฟ้มข้อมูลที่จัดเก็บข้อมูลตารางเวลาหรือการนัดหมายทั้งหมด ดังนั้นจึงเป็นแฟ้มที่มีขนาดใหญ่มากที่สุดและมีความสำคัญมากที่สุด โดยที่แต่ละเรคคอร์ดหรือแต่ละบรรทัดเท่ากับข้อมูลการนัดหมายหนึ่งอย่าง ซึ่งเรคคอร์ดของข้อมูลการนัดหมายนั้นได้ถูกแบ่งออกเป็น 5 ประเภท (ธรรมดา,รายวัน ,รายสัปดาห์,รายเดือน,รายปี) ดังนั้นรูปแบบการจัดเก็บข้อมูลในเรคคอร์ดของการนัดหมายแต่ละประเภทนั้นจะแตกต่างกัน แต่ก็มีบางฟิลด์ที่ซ้ำกัน ซึ่งมีอยู่ด้วยกันดังต่อไปนี้
  - **Record_Id** ทำหน้าที่ในการเก็บหมายเลขประจำเรคคอร์ด ซึ่งถูกสร้างขึ้นมาโดยใช้วันเดือนปีและเวลาบนเครื่องบริการ
  - **Appointment_Type** ทำหน้าที่ในการเก็บค่าที่บ่งบอกว่าเรคคอร์ดนี้เป็นข้อมูลการนัดหมายประเภทใด
  - **Start_Hour,Start_Min** ทำหน้าที่ในการเก็บชั่วโมงและนาทีของเวลาเริ่มต้นของการนัดหมาย
  - **End_Hour,End_Min** ทำหน้าที่ในการเก็บชั่วโมงและนาทีของเวลาสิ้นสุดของการนัดหมาย
  - **Where** ทำหน้าที่ในการเก็บข้อความที่บ่งบอกถึงสถานที่ที่จะเกิดการนัดหมายขึ้น

- Private ทำหน้าที่ในการเก็บข้อมูลที่บ่งบอกว่าต้องการให้ผู้ใช้คนอื่น (ที่ได้รับอนุญาต) มองเห็น Description ของการนัดหมายเรคคอร์ดนี้หรือไม่
- Remind ทำหน้าที่ในการเก็บข้อมูลที่บ่งบอกว่าต้องการให้ระบบเตือนความจำ (Reminder system) สำหรับการนัดหมายเรคคอร์ดนี้ทำงานหรือไม่ ซึ่งระบบนี้จะทำงานโดยที่เมื่อถึงเวลาการนัดหมายนั้นๆ โปรแกรมก็จะส่งข้อความเตือนผู้ใช้ ซึ่งระบบการเตือนความจำนี้สามารถกำหนดให้ทำงานล่วงหน้าได้โดยขึ้นอยู่กับพารามิเตอร์ Remind_Before_Time ในแฟ้ม config.cfg ที่จะถูกกล่าวถึงภายหลัง
- Description ทำหน้าที่เก็บข้อความคำอธิบายแบบสั้นของการนัดหมาย ฟิลด์นี้มีความสำคัญเพราะเป็นฟิลด์ที่ผู้ใช้คนอื่นสามารถมองเห็นได้
- Note ทำหน้าที่เก็บคำอธิบายแบบยาว

ส่วนฟิลด์ที่เหลือจะมีความแตกต่างไปตามประเภทของการนัดหมายซึ่งมีอยู่ด้วยกันดังต่อไปนี้

- <u>การนัดหมายประเภทธรรมดา (Normal)</u>
  - Date ทำหน้าที่เก็บวันเดือนปีที่เกิดการนัดหมาย
- <u>การนัดหมายประเภทรายวัน (Daily recurring)</u>
  - Start_Date ทำหน้าที่เก็บวันเดือนปีเริ่มต้นของการนัดหมายประเภทรายวัน
  - End_Date ทำหน้าที่เก็บวันเดือนปีสิ้นสุดของการนัดหมายประเภทรายวัน
  - Except_Date_List ทำหน้าที่เก็บลิสต์ (List) ของวันเดือนปีที่มีการยกเว้นการนัดหมายประเภทรายวัน
  - Every_Days ทำหน้าที่เก็บตัวเลขจำนวนวันที่จะเกิดการนัดหมาย 1 ครั้ง
- <u>การนัดหมายประเภทรายสัปดาห์ (Weely recurring)</u>
  - Start_Date ทำหน้าที่เก็บวันเดือนปีเริ่มต้นของการนัดหมายประเภทรายสัปดาห์

- End_Date ทำหน้าที่เก็บวันเดือนปีสิ้นสุดของการนัดหมายประเภทรายสัปดาห์
- Except_Date_List ทำหน้าที่เก็บลิสต์ของวันเดือนปีที่มีการยกเว้นการนัดหมายประเภทรายสัปดาห์
- Every_Weeks ทำหน้าที่เก็บตัวเลขบ่งบอกจำนวนสัปดาห์ที่เว้นช่วงในการเกิดการนัดหมาย
- Day_Of_Week_Pattern ทำหน้าที่เก็บค่าที่บ่งบอกว่าวันไหนในสัปดาห์บ้างนับตั้งแต่วันอาทิตย์จนถึงวันเสาร์

- <u>การนัดหมายประเภทรายเดือน (Monthly recurring)</u>
  - Start_Date ทำหน้าที่เก็บวันเดือนปีเริ่มต้นของการนัดหมายประเภทรายเดือน
  - End_Date ทำหน้าที่เก็บวันเดือนปีสิ้นสุดของการนัดหมายประเภทรายเดือน
  - Except_Date_List ทำหน้าที่เก็บลิสต์ของวันเดือนปีที่มีการยกเว้นการนัดหมายประเภทรายเดือน
  - Day_Of_Month ทำหน้าที่เก็บวันที่ที่เกิดการนัดหมายขึ้นในเดือนหนึ่งๆ
  - Rank_Of_Day ทำหน้าที่เก็บลำดับของวัน โดยจะถูกใช้ควบคู่กันกับ Day_Of_Week ตัวอย่างเช่น วันเสาร์แรกของเดือน เป็นต้น
  - Day_Of_Week ทำหน้าที่เก็บวันในแต่ละสัปดาห์ (อาทิตย์ - เสาร์) ซึ่งฟิลด์นี้จะถูกใช้ควบคู่กันกับ Rank_Of_Day
  - Every_Months ทำหน้าที่เก็บตัวเลขบ่งบอกอันดับของเดือนที่เว้นช่วงในการเกิดการนัดหมายแต่ละครั้ง
- <u>การนัดหมายประเภทรายปี (Yearly recurring)</u>
  - Start_Date ทำหน้าที่เก็บวันเดือนปีเริ่มต้นของการนัดหมายประเภทรายปี
  - End_Date ทำหน้าที่เก็บวันเดือนปีสิ้นสุดของการนัดหมายประเภทรายปี

- Except_Date_List ทำหน้าที่เก็บลิสต์ของวันเดือนปีที่มีการยกเว้นการนัดหมายประเภทรายปี
- Day_Of_Month ทำหน้าที่เก็บวันที่ที่เกิดการนัดหมายขึ้นในเดือนหนึ่งๆ
- Rank_Of_Day ทำหน้าที่เก็บลำดับของวัน โดยจะถูกใช้ควบคู่กันกับ Day_Of_Week ตัวอย่างเช่น วันเสาร์แรกของเดือน เป็นต้น
- Day_Of_Week ทำหน้าที่เก็บวันในแต่ละสัปดาห์ (อาทิตย์ - เสาร์) ซึ่งฟิลด์นี้จะถูกใช้ควบคู่กันกับ Rank_Of_Day
- Month_Of_Year ทำหน้าที่เก็บเดือนที่จะเกิดการนัดหมาย

- allow.db เป็นแฟ้มข้อมูลที่เก็บลิสต์ของ user name ที่ผู้ใช้มีสิทธิ์ดูตารางเวลาได้ ซึ่งแต่ละเรคคอร์ดในแฟ้มข้อมูลนี้จะมีอยู่เพียงฟิลด์เดียวคือ User_Name และข้อมูลที่อยู่ในแฟ้มนี้จะสัมพันธ์กันกับพารามิเตอร์ที่มีชื่อว่า Granted_User ใน config.cfg ที่จะถูกกล่าวถึงต่อไป
- config.cfg เป็นแฟ้มข้อมูลที่จัดเก็บพารามิเตอร์ต่างๆ ของผู้ใช้แต่ละคน ตัวอย่างเช่น เวลาเริ่มต้นและสิ้นสุดของการแสดงผลตารางเวลา สีของตัวอักษรและสีพื้นของการนัดหมายแต่ละประเภทบนตารางเวลาเป็นต้น ซึ่งพารามิเตอร์ต่างๆ ได้ถูกแบ่งออกเป็น 2 ประเภทดังต่อไปนี้
  - **พารามิเตอร์ส่วนของระบบ** เป็นพารามิเตอร์ที่ผู้ใช้ไม่สามารถเปลี่ยนแปลงได้โดยตรง ดังนั้นพารามิเตอร์เหล่านี้จะถูกแก้ไขโดยเครื่องบริการ ซึ่งมีอยู่ด้วยกันดังต่อไปนี้
    - Client_Address เก็บไอพีแอดเดรสของเครื่องใช้บริการที่ผู้ใช้ใช้ในการล็อกอิน
    - Client_Id เก็บรหัสประจำตัวของผู้ใช้แต่ละคนสำหรับการล็อกอินแต่ละครั้ง
    - Client_Still_Login เก็บสถานภาพการใช้งานของผู้ใช้ว่ายังล็อกอินอยู่หรือไม่
    - Last_Activate_Time เก็บเวลาที่ผู้ใช้มีการติดต่อกับเครื่องบริการครั้งล่าสุด

- Last_From_Address เก็บไอพีแอดเดรสของเครื่องใช้บริการที่ถูกใช้ในการล็อกอินครั้งก่อนหน้านี้
- Last_Login_Time เช่นเดียวกันกับ Last_From_Address แต่เก็บวันเวลาที่ผู้ใช้ล็อกอินครั้งก่อน

- **<u>พารามิเตอร์ส่วนของผู้ใช้</u>** เป็นพารามิเตอร์ที่ผู้ใช้แต่ละคนสามารถเปลี่ยนแปลงแก้ไขได้ตามต้องการ ซึ่งมีอยู่ด้วยกันดังต่อไปนี้
  - Start_View_Time เก็บค่าของชั่วโมงเริ่มต้นที่ใช้ในการแสดงตารางเวลา
  - End_View_Time เก็บค่าของชั่วโมงสิ้นสุดที่ใช้ในการแสดงตารางเวลา
  - Time_Scale เก็บค่าความละเอียดของการแสดงตารางเวลา
  - Remind_Before_Time เก็บค่าจำนวนนาทีที่เครื่องใช้บริการจะทำการเตือนความจำล่วงหน้าของการนัดหมาย
  - Normal_Foreground_Color เก็บค่าสีของตัวอักษรของการนัดหมายประเภทธรรมดา
  - Normal_Background_Color เก็บค่าสีของพื้นของการนัดหมายประเภทธรรมดา
  - Daily_Foreground_Color เก็บค่าสีของตัวอักษรของการนัดหมายประเภทรายวัน
  - Daily_Background_Color เก็บค่าสีของพื้นของการนัดหมายประเภทรายวัน
  - Weekly_Foreground_Color เก็บค่าสีของตัวอักษรของการนัดหมายประเภทรายสัปดาห์
  - Weekly_Background_Color เก็บค่าสีของพื้นของการนัดหมายประเภทรายสัปดาห์
  - Monthly_Foreground_Color เก็บค่าสีของตัวอักษรของการนัดหมายประเภทรายเดือน
  - Monthly_Background_Color เก็บค่าสีของพื้นของการนัดหมายประเภทรายเดือน

- Yearly_Foreground_Color เก็บค่าสีของตัวอักษรของการนัดหมายประเภทรายปี
- Yearly_Background_Color เก็บค่าสีของพื้นของการนัดหมายประเภทรายปี
- Other_Foreground_Color เก็บค่าสีของตัวอักษรของการนัดหมายของผู้ใช้คนอื่น
- Other_Background_Color เก็บค่าสีของพื้นของการนัดหมายของผู้ใช้คนอื่น
- Email เก็บค่าอีเมลล์แอดเดรสของผู้ใช้
- Granted_User เก็บลิสต์ของ user name ของผู้ใช้คนอื่นที่ผู้ใช้ให้สิทธิ์ในการดูตารางเวลาของตน ซึ่งพารามิเตอร์นี้จะถูกใช้ควบคู่กันกับแฟ้ม allow.db ตัวอย่างเช่น หาก john อนุญาตให้ david ดูตารางเวลาของตนเองได้ พารามิเตอร์ Granted_User ของ john จะมีชื่อ david อยู่ด้วย และ ในแฟ้ม allow.db ของ david จะมีชื่อ john อยู่ เป็นต้น
- HTML_Start_Date เป็นพารามิเตอร์ที่ถูกใช้ในตอนที่จะเปลี่ยนข้อมูลตารางเวลาให้เป็นแฟ้มข้อมูล HTML ซึ่งพารามิเตอร์ชนิดนี้จะขึ้นต้นด้วย "HTML_" ดังนั้นพารามิเตอร์นี้จะเก็บวันเดือนปีเริ่มต้นที่จะเปลี่ยนแปลงข้อมูลตารางเวลา
- HTML_End_Date เก็บวันเดือนปีสิ้นสุดที่จะเปลี่ยนแปลงข้อมูลตารางเวลา
- HTML_Start_Time เก็บชั่วโมงเริ่มต้นของการแปลง
- HTML_End_Time เก็บชั่วโมงสิ้นสุดของการแปลง
- HTML_Time_Scale เก็บความละเอียดของการแปลง
- HTML_Show_Detail เก็บค่าที่บ่งบอกว่าแฟ้ม HTML ที่เป็นผลของการแปลงนั้นจะแสดง Description หรือไม่ถ้าไม่จะแสดงคำว่า "Busy" แทน
- HTML_Text_Color เก็บค่าสีของตัวอักษรที่แสดงส่วนหัวของตารางเวลา

- HTML_Background_Image เก็บชื่อแฟ้มรูปภาพ JPEG หรือ GIF ที่เป็นพื้นหลัง
- HTML_Foreground_Color เก็บค่าของสีตัวอักษรบนส่วนที่เป็นการนัดหมายบนตารางเวลา
- HTML_Background_Color เก็บค่าของสีพื้นบนส่วนที่เป็นการนัดหมายบนตารางเวลา
- HTML_Table_Color เก็บค่าสีของตารางเวลา

3.3 การเข้าถึงแฟ้มข้อมูล (File Access)

การเข้าถึงข้อมูลทุกแฟ้มของโปรแกรมในส่วนเครื่องบริการเป็นแบบลำดับ (Sequential) ซึ่ง CSS เป็นโปรแกรมที่รองรับการทำงานของผู้ใช้หลายๆ คนพร้อมๆกัน แฟ้มข้อมูลหนึ่งอาจจะถูกใช้งานโดยผู้ใช้หลายๆคนพร้อมกัน ดังนั้นกลไกการล็อค (Lock) แฟ้มข้อมูลจึงเป็นสิ่งสำคัญ ซึ่งกลไกในการล็อคแฟ้มข้อมูลมีอยู่ด้วยกัน 2 ประเภท

3.3.1 Read lock คือการล็อคแฟ้มข้อมูลเพื่อการอ่าน การล็อคชนิดนี้สามารถเกิดขึ้นได้หลายๆ ครั้งพร้อมกันบนแฟ้มข้อมูลเดียวกัน

3.3.2 Write lock คือการล็อคแฟ้มข้อมูลเพื่อการเขียน การล็อคชนิดนี้สามารถทำได้ทีละครั้งบนแฟ้มข้อมูลหนึ่งๆ ไม่สามารถใช้งานร่วมกับ read lock ได้ในเวลาเดียวกัน

สำหรับฟังก์ชันสำหรับการล็อคทั้งสองแบบจะเป็นฟังก์ชันที่สามารถเรียกใช้ได้บนระบบยูนิกซ์

การล็อคแฟ้มข้อมูลนั้นมีขั้นตอนดังต่อไปนี้

```
openfile(FILEHANDLE,filename)
lockfile(FILEHANDLE)
... read/write data
unlockfile(FILEHANDLE)
closefile(FILEHANDLE)
```

จากขั้นตอนข้างต้นหาก lockfile ไม่สำเร็จ โปรแกรมก็จะหยุดรอจนกว่าจะสำเร็จจึงจะทำงานต่อ หากมองโดยรวมเพียงผิวเผินอาจจะไม่มีปัญหา แต่ถ้าสังเกตจะพบว่า การ lockfile จะเกิดขึ้นหลังจากการ openfile และการ unlockfile จะเกิดขึ้นก่อนการ closefile หรือกล่าวได้อีกอย่างว่าเกิดการใช้งานแฟ้มข้อมูลก่อนที่จะล็อคแฟ้มข้อมูลนั้นๆ และปลดล็อคก่อนที่จะสิ้นสุดการใช้งานกับแฟ้มข้อมูล ถ้าเป็นการล็อคเพื่อการอ่านอาจจะไม่เป็นปัญหา แต่ปัญหาจะเกิดตอนล็อคเพื่อการเขียน นั่นคือการแก้ไขเปลี่ยนแปลงข้อมูลลงบนดิสต์ (Disk) จะเกิดขึ้นในขั้นตอนการ closefile

แต่ทว่า ก่อนที่จะเกิดการ closefile โปรแกรมได้ทำการ unlockfile ไปแล้ว ดังนั้นจึงมีความเป็นไปได้ที่จะมีโปรแกรมอื่นที่รอการใช้งานแฟ้มดังกล่าวอยู่และเข้ามาล็อคเพื่อเข้าถึงข้อมูลก่อนการ closefile ซึ่งจะก่อให้เกิดความผิดพลาดในการอ่านหรือเขียนข้อมูลครั้งหลังได้

เพราะฉะนั้นผู้วิจัยจึงมีแนวคิดในการแก้ไขโดยการเพิ่มแฟ้มข้อมูลสำหรับการล็อคโดยเฉพาะ ตัวอย่างเช่นหากต้องการอ่านเขียนบนแฟ้มข้อมูล appointment.db การล็อคจะไม่เกิดขึ้นกับแฟ้มข้อมูลนี้โดยตรง แต่จะไปเกิดขึ้นกับ appointment.db.lock แทน ซึ่งเป็นแฟ้มข้อมูลว่างเปล่าที่ถูกใช้เพื่อการล็อคโดยเฉพาะ สามารถเขียนเป็นโคดได้ดังต่อไปนี้

```
openfile(LOCKFILE,appointment.db.lock)
lockfile(LOCKFILE)
openfile(FILEHANDLE,appointment.db)
... read/write data on appointment.db
closefile(FILEHANDLE)
unlockfile(LOCKFILE)
closefile(LOCKFILE)
```

จะเห็นได้ว่าปัญหาดังกล่าวได้ถูกขจัดไป เนื่องจากการล็อคเกิดขึ้นก่อนที่จะใช้งานแฟ้ม appointment.db และปลดล็อคหลังจากสิ้นสุดการใช้งาน appointment.db ซึ่งการล็อคด้วยวิธีนี้สามารถใช้ได้ทั้งการล็อคเพื่อการอ่านและเขียน

3.4 การทำงานของโปรแกรม kernel

จากที่ได้กล่าวไปแล้วว่าแฟ้มข้อมูลโปรแกรม kernel ที่อยู่ในไดเรกทอรี /user/schedule/bin นั้นได้ถูกพัฒนาออกมาในลักษณะ Fast CGI ซึ่งมีฟังก์ชันการทำงานหลักๆ ดังต่อไปนี้

3.4.1 **Add_New_Account** เป็นฟังก์ชันที่ทำหน้าที่ในการสร้าง account ของผู้ใช้เรคคอร์ดใหม่ โดยจะรับค่าอินพุท คือ user name และ pasword มาเพื่อทำการสร้างไดเรกทอรีสำหรับผู้ใช้รวมทั้งแฟ้มข้อมูล allow.db appointment.db และ config.cfg ภายใต้ไดเรกทอรีนั้นๆ

3.4.2 **Main** เป็นฟังก์ชันที่ทำหน้าที่ในการตรวจสอบการล็อกอินของผู้ใช้ โดยจะรับค่าอินพุทเป็น user name และ password มา โดยเริ่มแรกจะทำการตรวจสอบกับพารามิเตอร์ Enable_Login ในแฟ้ม server.cfg ว่าอนุญาตให้มีการล็อกอินหรือไม่ จากนั้นก็จะทำการตรวจสอบ user name และ password ดังกล่าวกับเรคคอร์ดในแฟ้มข้อมูล account_*.db ว่าตรงกันหรือไม่ หากถูกต้องจึงอนุญาตให้ผู้ใช้เริ่มใช้งาน

โปรแกรมได้ ขั้นตอนต่อมาก็จะทำการส่ง HTML page ไปยังเครื่องใช้บริการ ซึ่งใน HTML หน้านี้จะเป็นส่วนหลักที่ควบคุมการประสานกับผู้ใช้ ซึ่งผู้วิจัยขอเรียก HTML หน้านี้ว่า "main page" โดยมีส่วนประกอบดังต่อไปนี้

- javascript ที่ทำการรับค่าของพารามิเตอร์ที่อยู่ใน config.cfg มายังตัวแปร javascript เพื่อให้เครื่องใช้บริการรับทราบค่าพารามิเตอร์ต่างๆของเครื่องบริการ
- javascript ที่ทำการประกาศตัวแปรต่างๆ ที่ถูกใช้ร่วมกัน (Global) บนเครื่องใช้บริการ
- รายชื่อของแฟ้มข้อมูล HTML ต่างๆ ที่ถูกบรรจุอยู่ในเฟรม (Frame) บน main page

ขั้นตอนสุดท้ายฟังก์ชันนี้จะทำการสร้าง Client_Id สำหรับการล็อกอินครั้งนี้และจำค่าไอพีแอดเดรสของเครื่องใช้บริการเพื่อเขียนลงไปใน config.cfg ของผู้ใช้คนนั้นๆ ซึ่ง Client_Id จะถูกใช้เป็นอินพุทของฟังก์ชันตั้งแต่ข้อ 3.4.3 ขึ้นไป เพื่อที่ฟังก์ชันเหล่านี้จะสามารถแยกแยะได้ว่าจะจัดการกับแฟ้มข้อมูลของผู้ใช้คนใด

3.4.3 Get_Reminder_Appointment เป็นฟังก์ชันที่ทำหน้าที่ในการส่งข้อมูลการนัดหมายไปยังเครื่องใช้บริการเพื่อใช้ในระบบเตือนความจำ โดยรับอินพุทเป็น Client_Id และวันเดือนปีต้องการเตือน โดยจะทำการตรวจสอบกับ account.db และนำเอาเรคคอร์ดที่ตรงตามเงื่อนไขและส่งข้อมูลสู่เครื่องใช้บริการในลักษณะของโปรแกรม javascript ซึ่งการส่งข้อมูลเป็นแบบโปรแกรม Javascript จะถูกนำเสนออีกครั้งในบทที่ 5

3.4.4 Get_Daily_Appointment เป็นฟังก์ชันที่ทำหน้าที่ในการส่งข้อมูลการนัดหมายสำหรับวันเดือนปีใดๆ ให้กับเครื่องใช้บริการ โดยรับอินพุทเป็น Client_Id และ วันเดือนปี จากนั้นก็จะทำการตรวจสอบกับ account.db เพื่อดึงเอาเรคคอร์ดของการนัดหมายเฉพาะวันนั้นๆ ออกมา และส่งให้เครื่องใช้บริการในลักษณะของโปรแกรม javascript

3.4.5 Get_Weekly_Appointment เป็นฟังก์ชันที่ทำหน้าที่ในการส่งผ่านข้อมูลการนัดหมายสำหรับสัปดาห์ใดๆ ให้กับเครื่องใช้บริการ โดยรับอินพุทเป็น Client_Id และ วันเดือนปี จากนั้นก็จะทำการตรวจสอบกับ account.db เพื่อดึงเอาเรคคอร์ดของการนัดหมายเฉพาะสัปดาห์นั้นๆ ออกมา และส่งให้เครื่องใช้บริการในลักษณะของโปรแกรม javascript

3.4.6 Get_List_Appointment เป็นฟังก์ชันที่ทำหน้าที่ส่งผ่านข้อมูลการนัดหมายไปยังเครื่องใช้บริการในลักษณะโปรแกรม javascript โดยจะส่งไปครั้งละ 20 เรคคอร์ด โดยเรียงลำดับจากวันเดือนปีและประเภทของการนัดหมาย

3.4.7 Get_Range_Appointment เป็นฟังก์ชันที่ทำหน้าที่ในการส่งผ่านข้อมูลการนัดหมายในช่วงระยะเวลาที่กำหนดไปยังเครื่องใช้บริการในลักษณะโปรแกรม javascript ซึ่งสามารถกำหนดช่วงเวลาได้ตั้งแต่ 1 วันถึง 2 เดือน โดยฟังก์ชันจะรับค่าอินพุทเป็น Client_Id วันเดือนปีเริ่มต้น และ วันเดือนปีสิ้นสุด โดยทำการตรวจสอบกับแฟ้มข้อมูล account.db เพื่อดึงข้อมูลการนัดหมายตามช่วงเวลาที่กำหนด

3.4.8 Modify_Appointment เป็นฟังก์ชันที่ทำหน้าที่ในการสร้าง แก้ไข และลบข้อมูลการนัดหมาย โดยฟังก์ชันนี้จะรับค่าอินพุทเป็น Client_Id และฟิลด์ต่างๆ ของการนัดหมายแต่ละประเภท พร้อมกับค่าที่บ่งบอกว่าต้องการให้ฟังก์ชันนี้ทำการสร้าง แก้ไขหรือลบเรคคอร์ดของการนัดหมาย

3.4.9 Change_Password เป็นฟังก์ชันที่ทำหน้าที่ในการเปลี่ยน password ของการล็อกอิน โดยจะรับอินพุทเป็น password เก่าและ password ใหม่ จากนั้นฟังก์ชันนี้ก็จะทำการตรวจสอบว่า password เก่านั้นถูกต้องหรือไม่ ถ้าหากถูกจึงยอมให้มีการเปลี่ยนเป็น password ใหม่ได้

3.4.10 Delete_Account เป็นฟังก์ชันสำหรับการลบ account ที่กำลังใช้งานอยู่ ซึ่งจะถูกใช้ในกรณีที่ผู้ใช้ไม่ต้องการ account นี้ของตนเองอีกต่อไป โดยจะทำการลบแฟ้มข้อมูล allow.db, config.cfg, appointment.db และไดเรกทอรีของผู้ใช้คนนั้นๆ รวมทั้งลบเรคคอร์ดใน account_*.db ของผู้ใช้คนนั้นๆ ออกด้วย

3.4.11 Client_Logout เป็นฟังก์ชันสำหรับการล็อกเอ้าท์ (Logout) ของผู้ใช้ ซึ่งเป็นขั้นตอนที่ผู้ใช้ทุกคนควรทำก่อนสิ้นสุดการใช้งานเครื่องใช้บริการ เนื่องจากพารามิเตอร์ต่างๆ ที่ผู้ใช้ได้ตั้งไว้จะถูกบันทึกลงในแฟ้ม config.cfg ในฟังก์ชันนี้ ดังนั้นอินพุทคือ Client_Id และพารามิเตอร์ส่วนของผู้ใช้ทั้งหมดที่อยู่ใน config.cfg

จากฟังก์ชันทั้งหมดที่ได้กล่าวมาข้างต้น ทุกฟังก์ชันจะมีการตรวจสอบอินพุทที่รับเข้ามาก่อนที่จะนำไปใช้งานว่าค่าต่างๆ นั้นอยู่ในขอบเขตที่กำหนดไว้หรือไม่ สาเหตุที่ต้องตรวจสอบเนื่องจากโปรแกรมไม่ว่าจะเป็น CGI หรือ Fast CGI ก็ตาม เป็นโปรแกรมที่ถูกเรียกใช้งานโดยเครื่องใช้บริการที่อยู่ห่างไกล และข้อมูลอินพุทที่โปรแกรมได้รับมานั้นก็เป็นสิ่งที่เครื่องใช้บริการเป็นผู้ส่งมาด้วยเช่นกัน ดังนั้นหากไม่มีการกลั่นกรองอินพุทเหล่านี้อย่างละเอียดถี่ถ้วนแล้วจะเป็นสิ่งที่อันตรายในกรณีที่ผู้ไม่หวังดีใช้ช่องโหว่ดังกล่าวเจาะเข้ามาในเครื่องบริการที่โปรแกรมทำงานอยู่ ดังนั้นการตรวจสอบอินพุทของทุกฟังก์ชันก่อนการใช้งานจึงเป็นสิ่งที่สำคัญอย่างยิ่ง ตัวอย่างเช่น อินพุทที่

เป็นเลขชั่วโมง ซึ่งมีค่าที่เป็นไปได้อยู่ในช่วง 0 - 23 ดังนั้นหากได้รับค่าอื่นที่นอกเหนือจากนี้มา ฟังก์ชันก็จะปฏิเสธการร้องขอครั้งนั้นๆ เป็นต้น

สาเหตุที่ฟังก์ชันทั้งหมดได้ถูกเก็บรวมเข้าไปอยู่ในแฟ้มโปรแกรม kernel เพียงแฟ้มเดียว เพราะโปรแกรม Fast CGI เป็นโปรแกรมที่ถูกเรียกขึ้นมาเป็นเดมอนโปรเซส หากฟังก์ชันทั้งหมดถูกจับรวมเข้าไปใน kernel แล้ว ก็จะมีเดมอนโปรเซส kernel เพียงโปรเซสเดียวเท่านั้น แต่ถ้าหากแยกฟังก์ชันต่างๆ ไปอยู่คนละแฟ้ม ดังนั้นฟังก์ชันทั้ง 11 ฟังก์ชันก็จะมีโปรแกรม Fast CGI 11 แฟ้ม เป็นผลให้จะต้องมีเดมอนโปรเซสเป็นจำนวนถึง 11 โปรเซส และทรัพยากรของระบบ (System resource) ที่ถูกใช้งานทั้ง 11 โปรเซสรวมกันจะมากกว่า kernel โปรเซสเดียว เนื่องจากทั้ง 11 โปรเซสนั้นต้องการทรัพยากรระบบเป็นของตัวเองและทรัพยากรระบบเหล่านี้ของทั้ง 11 โปรเซสส่วนใหญ่จะซ้ำซ้อนกันแต่ไม่สามารถใช้งานร่วมกันได้ ในขณะที่ kernel ซึ่งรวมฟังก์ชันทั้งหมดเอาไว้ด้วยกัน เป็นผลทำให้ฟังก์ชันเหล่านี้สามารถใช้ทรัพยากรของระบบร่วมกันได้