บทที่ 8
การจัดการไฟล์และไดเรกทอรี

การจัดเก็บข้อมูลของคอมพิวเตอร์จะทำในรูปแบบของไฟล์ ทั้งนี้ในกระบวนการสร้างเว็บในบางครั้งมีความจำเป็นต้องใช้ความรู้เรื่องการจัดการไฟล์ (File) และไดเรกทอรี (Directory) มาเกี่ยวข้อง ดังนั้นในบทนี้จึงจะกล่าวถึงเรื่องราวพื้นฐานที่จำเป็นที่ควรรู้เกี่ยวกับไฟล์และไดเรกทอรี เช่น การอ้างอิงตำแหน่ง การสร้าง การลบ การเปลี่ยนชื่อ การเคลื่อนย้าย การเปิด และการอ่าน เป็นต้น

8.1 การอ้างอิงตำแหน่งไฟล์และไดเรกทอรี

ในการสร้างเว็บเพจนั้น บางครั้งจำเป็นต้องนำข้อมูลจากไฟล์ต่างๆ เข้ามาใช้ร่วมด้วย เช่น รูปภาพ หรือการนำไฟล์เข้ามาแทรก ทั้งนี้ไฟล์แต่ละชนิด มักแยกเก็บให้เป็นสัดส่วนเฉพาะ เช่น ไฟล์ภาพ อาจเก็บไว้ในไดเรกทอรี images หรือ pictures หรือไฟล์สำหรับนำไปแทรกจะเก็บไว้ในไดเรกทอรี inc เป็นต้น ด้วยเหตุนี้จึงมักเกิดปัญหาในการอ้างอิง ตำแหน่งไฟล์ที่อยู่ต่างไดเรกทอรี ดังนั้นก่อนที่จะศึกษาเรื่องอื่นๆ จะต้องทำความเข้าใจเกี่ยวกับการอ้างอิงตำแหน่งของไฟล์และไดเรกทอรีก่อน มีรายละเอียด ดังนี้

8.1.1 การอ้างถึง Document Root

Document Root คือ ไดเรกทอรีเริ่มต้นของการจัดเก็บข้อมูลของเว็บไซต์ ตำแหน่งนี้ปกติแล้วจะขึ้นอยู่กับการตั้งค่าและปรับแต่งระบบโดยผู้ดูแลระบบ ทั้งนี้หากเป็นผู้ติดตั้งระบบเองด้วย AppServ ตามค่าปกติจะกำหนดที่อยู่ไว้ ดังนี้

```
C:/AppServ/www
```

สำหรับตำแหน่งนี้ถ้าเป็นรูปแบบ URL จะเท่ากับ http://localhost หรือ http://127.0.0.1 นั้นเอง ตามรูปแบบของระบบเว็บจะแทนตำแหน่งนี้ด้วยเครื่องหมาย / แล้วตามด้วยตำแหน่งอ้างอิงตำแหน่งไฟล์ และไดเรกทอรี เป็นต้นไป เช่น “/myphp/example/index.php”

8.1.2 การอ้างอิงพาธแบบ Absolute

พาธ คือ เส้นทางไปยังไฟล์หรือไดเรกทอรี การอ้างอิงแบบ Absolute จะเป็นการระบุตำแหน่งที่แท้จริงของไฟล์ หรือไดเรกทอรี โดยเริ่มจาก Document Root เป็นต้นไป

ลักษณะการอ้างอิงไฟล์ภาพต่างๆ จะเป็นดังนี้

```
<img src = “/myphp/img/logo.png” />
<a href = “/myphp/example/login.php “>เข้าสู้ระบบ < /a>
```

จะเห็นว่าการอ้างอิงพาธแบบ Absolute นี้ต้องไปเริ่มจาก Document Root เสมอ (เริ่มพาธด้วยเครื่องหมาย /) แล้วตามด้วยไดเรกทอรีตามลำดับจนถึงไฟล์ที่ต้องการ ไม่ว่าจะนำไฟล์ไปใช้ที่เว็บเพจ

ใดก็ตาม เพราะก็จะอ้างอิงจากตำแหน่ง Document Root เสมอ อย่างไรก็ตามการระบุพาธแบบ Absolute นี้ จะใช้ได้ผลเฉพาะกับ HTML tags เท่านั้น หากใช้ร่วมกับฟังก์ชันต่างๆ ของระบบไฟล์ด้วยภาษา PHP จะเกิดปัญหา เว้นแต่จะระบุตำแหน่ง Document Root ก่อนการเรียกใช้งาน ตัวอย่างดังนี้

ตัวอย่างที่ 8.1 การระบุตำแหน่ง Document Root ก่อนการใช้งาน

```
$root = $_SERVER ['DOCUMENT_ROOT'] ;
$path = "myphp/example/inc/header.inc.php";
include (“$root . $path”);
//ลักษณะผลลัพธ์ C:/AppServ/www/myphp/example/inc/header.inc.php
```

8.1.3 การอ้างอิงพาธแบบ Relative

การอ้างอิงพาธแบบ Relative จะอ้างอิงโดยการเทียบกับตำแหน่งไฟล์ที่เป็นผู้เรียกไฟล์อื่นมาใช้เป็นหลัก โดยมีสัญลักษณ์การเปรียบเทียบตำแหน่งเพิ่มเติมเข้ามาอีก 2 รูปแบบ ดังนี้

1) การใช้สัญลักษณ์ . (จุดหนึ่งจุด) ใช้แทนไดเรกทอรีปัจจุบัน

2) การใช้สัญลักษณ์ .. (จุดสองจุด) ใช้แทนไดเรกทอรีก่อนหน้านี้ 1 ระดับ เช่น หากปัจจุบันอยู่ที่ไดเรกทอรี /myphp/example ถ้าใช้ .. จะหมายถึงการอ้างอิงไปยังตำแหน่งไดเรกทอรี /myphp เป็นต้น

8.2 การจัดการกับไฟล์

การจัดเก็บข้อมูลลงในไฟล์ทำได้ 2 วิธี คือ 1) จัดเก็บในรูปของแฟลตไฟล์ (Flat File) หรือเท็กซ์ไฟล์ (Text File) และ 2) จัดเก็บในรูปแบบของฐานข้อมูล (Database) แต่ไม่ว่าจะมีการจัดเก็บแบบใดก็ตาม จำเป็นต้องมีการเขียน และอ่านข้อมูลในไฟล์เสมอ โดยมีขั้นตอนดังนี้

ตารางที่ 8.1 ขั้นตอนการเขียนและอ่านข้อมูลในไฟล์

การทำงาน	ขั้นตอน
การเขียน (Writing)	1. เปิดไฟล์ที่ต้องการหรือสร้างไฟล์ใหม่ 2. เขียนข้อมูลลงไฟล์ 3. ปิดไฟล์

การทำงาน	ขั้นตอน
การอ่าน (Reading)	1. เปิดไฟล์ 2. อ่านข้อมูลจากไฟล์ 3. ปิดไฟล์

ในหัวข้อนี้จะกล่าวถึงการจัดการกับไฟล์ประเภทเท็กซ์ไฟล์ มีฟังก์ชันในการจัดการดังต่อไปนี้

8.2.1 การเปิดไฟล์

การอ่านข้อมูลจากไฟล์หรือบันทึกข้อมูลลงไฟล์ จะต้องเริ่มด้วยการเปิดไฟล์ขึ้นมาก่อนเสมอ โดยใช้ฟังก์ชัน fopen ( ) สำหรับการเปิดไฟล์เพื่อการทำงานใดๆ นั้น สามารถระบุได้ในส่วนของโหมดไฟล์ (File Mode) ถ้าหากสามารถเปิดไฟล์ได้จะคืนค่าเป็นตำแหน่งของตัวชี้ตำแหน่งที่อยู่ใน

ไฟล์ข้อมูล (Resource) หากเปิดไม่ได้ หรือไม่พบไฟล์ที่ระบุ จะคืนค่าเป็นเท็จ รูปแบบการใช้งานฟังก์ชัน fopen ( ) ดังนี้

รูปแบบ

```
resource fopen ( string $filename, string $mode [, bool $use_include_path = false] )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่ต้องการเปิด

$mode หมายถึง โหมดไฟล์ ใช้ระบุว่าเปิดไฟล์เพื่อวัตถุประสงค์ใด (ดูค่าของโหมดไฟล์ได้จากตารางที่ 8.2)

$use_include_path หมายถึง หากกำหนดพารามิเตอร์เป็น 1 หรือ True ฟังก์ชันจะค้นหาไฟล์จากไดเรกทอรีที่กำหนดใน "include_path" (กำหนดค่าในไฟล์ php.ini)

ตารางที่ 8.2 โหมดไฟล์

โหมดไฟล์	ความหมาย
r	โหมดอ่าน – เปิดไฟล์สำหรับการอ่าน เริ่มทำงานจากตอนต้นไฟล์
r+	โหมดอ่าน – เปิดไฟล์สำหรับการอ่านและเขียน เริ่มทำงานจากตอนต้นไฟล์
w	โหมดเขียน – เปิดไฟล์สำหรับการเขียน เริ่มทำงานจากตอนต้นไฟล์ ถ้าไฟล์มีอยู่แล้วจะลบข้อมูลที่มีอยู่ ถ้าไม่มีไฟล์จะสร้างขึ้นใหม่
w+	โหมดเขียน – เปิดไฟล์สำหรับการเขียนและอ่าน เริ่มทำงานจากตอนต้นไฟล์ ถ้าไฟล์มีอยู่แล้วจะลบข้อมูลที่มีอยู่ ถ้าไม่มีไฟล์จะสร้างขึ้นใหม่
a	โหมดเพิ่ม – เปิดไฟล์สำหรับการเพิ่ม (เขียน) เริ่มทำงานจากจุดสิ้นสุดของข้อมูลที่มีอยู่ ถ้าไม่มีไฟล์จะได้รับการสร้างขึ้นใหม่
a+	โหมดเพิ่ม – เปิดไฟล์สำหรับการเพิ่ม (เขียน) และอ่าน เริ่มทำงานจากจุดสิ้นสุดของข้อมูลที่มีอยู่ ถ้าไม่มีไฟล์จะได้รับการสร้างขึ้นใหม่
b	โหมดไบนารี – ใช้ร่วมกับโหมดอื่น โดยอาจจะจำเป็น ถ้าระบบไฟล์แยกระหว่างไฟล์ไบนารีและเท็กซ์ไฟล์ ระบบ Windows มีการแยก แต่ระบบ Unix ไม่มีการแยก
t	ใช้ร่วมกับโหมดอื่นๆ ที่กล่าวมาข้างต้น เพื่อเป็นการเปิดไฟล์ในโหมดเท็กซ์ไฟล์ธรรมดา

หมายเหตุ

ตัวชี้ตำแหน่งข้อมูลในไฟล์ (File Pointer) หมายถึง ตัวแปรที่กำหนดขึ้นเพื่อใช้สำหรับการชี้ตำแหน่งข้อมูลในไฟล์ เมื่อเปิดไฟล์ ตัวชี้ตำแหน่งไฟล์ข้อมูลจะถูกกำหนดให้ชี้ไปยังข้อมูลแรกในไฟล์ โดยทุกครั้งหลังจากการอ่านหรือเขียนข้อมูลในไฟล์ ตัวชี้จะเลื่อนไปยังข้อมูลที่อยู่ตำแหน่งถัดไปโดยอัตโนมัติ

ตัวอย่างที่ 8.2 การเปิดไฟล์เพื่อเขียน

```
1    <?php
2        $fp = fopen ("test.txt", "w");
3    ?>
```

จากตัวอย่างที่ 8.2 การเปิดไฟล์เพื่อเขียนมีการกำหนดโหมดเป็น “w” เป็นโหมดสำหรับเปิดไฟล์เพื่อเขียน โดยถ้าไฟล์มีอยู่แล้วจะลบข้อมูลทั้งหมด ถ้าไม่มีไฟล์จะสร้างไฟล์ขึ้นใหม่ แต่หากต้องการเพิ่มข้อมูลต่อท้ายเรื่อยๆ จะต้องเปลี่ยนไฟล์โหมดตามตัวอย่างที่ 8.3

ตัวอย่างที่ 8.3 การเปิดไฟล์เพื่อเขียนต่อท้าย

```
1    <?php
2        $fp = fopen ("test.txt", "a");
3    ?>
```

จากตัวอย่างที่ 8.2 และ 8.3 กำหนดให้ตัวแปร $fp ทำหน้าที่เป็นตัวชี้ตำแหน่งข้อมูลในไฟล์ ซึ่งเป็นผลมาจากการใช้ฟังก์ชัน fopen ( ) หากไฟล์ที่ต้องการเปิดอยู่คนละไดเรกทอรี จะต้องระบุไดเรกทอรีของไฟล์ที่ต้องการเปิดด้วย เช่น "C:/AppServ/www/file/test.txt" โดย "test.txt" คือ ชื่อไฟล์ ส่วน "C:/AppServ/www/file/" คือ ไดเรกทอรีที่เก็บไฟล์ "test.txt" นอกจากนี้ยังอ้างถึงไฟล์ใน Internet/Intranet ได้ โดยระบุเป็น URL เช่น "http://www.freebsd.sru.ac.th" หรือเรียกไฟล์ผ่านโปรโตคอล FTP เช่น ftp://user:password@www.freebsd.sru.ac.th/file/test.txt เป็นต้น

ปัญหาที่อาจจะเกิดขึ้นจากการเปิดไฟล์หรือเขียนไฟล์ด้วย fopen ( ) คือ ผู้ใช้อาจไม่มีสิทธิ์ในการอ่านหรือเขียนข้อมูลในไฟล์ที่เซิร์ฟเวอร์ ส่วนใหญ่จะพบในระบบปฏิบัติการ Unix หรือ Linux เนื่องจากมีการกำหนดสิทธิการใช้งานของไฟล์แต่ละไฟล์ไว้ ดังนั้นถ้าใช้ฟังก์ชัน fopen ( ) แล้วเกิด error ในลักษณะของ “Permission denied” ให้ตรวจสอบสิทธิ์ในการใช้งานไฟล์อีกครั้งว่าสามารถใช้งานได้หรือไม่

ในกรณีที่ไม่สามารถใช้ฟังก์ชัน fopen ( ) เพื่อเปิดไฟล์ข้อมูลได้ สามารถเขียนคำสั่งเพื่อแจ้งข้อผิดพลาด โดยใช้โครงสร้างเงื่อนไข if ในการตรวจสอบข้อผิดพลาดที่อาจเกิดขึ้นได้ ดังนี้

ตัวอย่างที่ 8.4 การใช้โครงสร้างเงื่อนไข if ในการตรวจสอบข้อผิดพลาดเมื่อเปิดไฟล์ข้อมูลไม่ได้

```
1    <?php
2        $fp = @open (" /usr/local/apache/data/doc/customer.txt ". " w ")
3        If (!$fp) {
4            echo " ไม่สามารถเปิดไฟล์ได้ ";
5            exit;
6        }
7    ?>
```

จากตัวอย่างที่ 8.4 การใช้โครงสร้างเงื่อนไข if ในการตรวจสอบข้อผิดพลาดเมื่อเปิดไฟล์ข้อมูลไม่ได้ มีการระบุ @ ไว้หน้าฟังก์ชัน fopen ( ) เพื่อไม่ให้ฟังก์ชันแสดงข้อผิดพลาดของระบบขึ้นมาในกรณีที่ไม่สามารถเปิดไฟล์ได้ แต่จะใช้โครงสร้างเงื่อนไข if ในการตรวจสอบผลลัพธ์ที่ได้จากการเปิดไฟล์ด้วยผู้พัฒนาเอง เป็นต้น

8.2.2 การปิดไฟล์

ทุกครั้งที่มีการเปิดไฟล์ เมื่อใช้ทำงานตามวัตถุประสงค์เสร็จ หลังจากนั้นควรจะปิดไฟล์ทุกครั้ง โดยใช้ฟังก์ชัน fclose ( ) เพื่อปิดการเชื่อมต่อและคืนค่าหน่วยความจำกลับสู่ระบบ โดยเมื่อเรียกใช้ฟังก์ชัน fclose ( ) หากสามารถปิดไฟล์ได้ ฟังก์ชันจะคืนค่าเป็น true และหากปิดไฟล์ไม่สำเร็จ จะคืนค่าเป็น false รูปแบบการใช้งานฟังก์ชัน มีดังนี้

รูปแบบ

```
bool fclose ( resource $handle )
```

เมื่อ $handle หมายถึง Resource หรือตำแหน่งของตัวชี้ที่อยู่ในไฟล์ข้อมูล เป็นค่าที่ส่งคืนมาจากฟังก์ชัน fopen ( )

ตัวอย่างที่ 8.5 การใช้ฟังก์ชัน fclose ( )

```
1   <?php
2       $handle = fopen ("somefile.txt", "r");
3       fclose ($handle);
4   ?>
```

8.2.3 การเขียนข้อมูลลงในไฟล์

การเขียนข้อมูลลงในไฟล์ สามารถประยุกต์ใช้ได้ 3 ฟังก์ชันหลักๆ ประกอบด้วย 1) fwrite ( ) 2) fputs ( ) และ 3) file_put_content ( ) โดยก่อนจะเขียนข้อมูลลงในไฟล์จะต้องเปิดไฟล์ด้วยฟังก์ชัน fopen ( ) ก่อนเสมอ รายละเอียดดังนี้

1) ฟังก์ชัน fwrite ( ) เป็นฟังก์ชันสำหรับเขียนข้อมูลลงในไฟล์ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
int fwrite ( resource $handle , string $string [, int $length ] )
```

เมื่อ $handle หมายถึง Resource หรือตำแหน่งของตัวชี้ที่อยู่ในไฟล์ข้อมูล
$string หมายถึง ข้อมูลหรือข้อความที่ต้องการจัดเก็บลงไฟล์
$length หมายถึง จำนวนตัวอักษรที่ต้องการบันทึกลงไฟล์ (มีหน่วยเป็นไบต์) หากไม่กำหนดจะบันทึกกลุ่มตัวอักษรทั้งหมดจากตัวแปร $string ลงในไฟล์

ตัวอย่างที่ 8.6 การใช้ฟังก์ชัน fwrite ( )

```
1    <?php
2        $path = "C:/AppServ/www/file";
3        $filename = "testfwrite.txt";
4        $file = fopen ( $path . "/" . $filename, "w");
5        $data = "ทดสอบการเขียนข้อมูลลงไฟล์ด้วยฟังก์ชัน fwrite()";
6        if (fwrite ($file, $data)==True) {
7            echo "เขียนข้อมูลลงไฟล์เรียบร้อยแล้ว";
8        }
9        fclose ($file);
10   ?>
```

จากตัวอย่างที่ 8.6 การใช้ฟังก์ชัน fwrite ( ) เพื่อเขียนข้อมูลลงในไฟล์ อธิบายดังนี้

บรรทัดที่ 2 กำหนดให้ตัวแปร $path มีค่าเท่ากับ "C:/AppServ/www/file" (โดยความหมายแล้วหมายถึงการกำหนดตำแหน่งของ Document Root เพื่อใช้อ้างอิงตำแหน่งสำหรับเก็บไฟล์ข้อมูล)

บรรทัดที่ 3 กำหนดให้ตัวแปร $filename มีค่าเท่ากับ "testfwrite.txt" (ชื่อของเท็กซ์ไฟล์ที่ต้องการเขียนข้อมูล)

บรรทัดที่ 4 กำหนดให้ตัวแปร $file มีค่าเท่ากับ เรียกใช้ฟังก์ชัน fopen ( ) โดยระบุหรืออ้างอิงตำแหน่งของ Document Root ระบุเท็กซ์ไฟล์ที่ต้องการใช้งานและโหมดไฟล์ที่ต้องการใช้งาน คือ w (กำหนดให้ตัวแปร $file ทำหน้าที่เป็นตัวชี้ตำแหน่งข้อมูลในไฟล์)

บรรทัดที่ 5 กำหนดให้ตัวแปร $data มีค่าเท่ากับข้อความ "ทดสอบการเขียนข้อมูลลงไฟล์ด้วยฟังก์ชัน fwrite ( )"

บรรทัดที่ 6 ใช้โครงสร้างเงื่อนไข if ตรวจสอบผลการทำงานของฟังก์ชัน fwrite ( ) โดยระบุพารามิเตอร์ประกอบด้วย $file (ทำหน้าที่ตัวชี้ตำแหน่งข้อมูลในไฟล์) และ $data (คือ ข้อมูลที่จะใช้เขียนลงในไฟล์) กรณีผลการตรวจสอบมีค่าเป็น true ไปบรรทัดที่ 7

บรรทัดที่ 7 แสดงข้อความ "เขียนข้อมูลลงไฟล์เรียบร้อยแล้ว"

บรรทัดที่ 8 ปิดการเชื่อมต่อกับไฟล์

2) ฟังก์ชัน fputs ( ) เป็นฟังก์ชันที่ใช้สำหรับเขียนข้อมูลลงไฟล์ โดยการเรียกใช้ และการทำงานจะเหมือนกับฟังก์ชัน fwrite ( ) มีรูปแบบดังนี้

รูปแบบ

```
int fputs ( resource $handle , string $string [, int $length ] )
```

เมื่อ $handle หมายถึง Resource หรือตำแหน่งของตัวชี้ที่อยู่ในไฟล์ข้อมูล

$string หมายถึง ข้อมูลหรือข้อความที่ต้องการจัดเก็บลงไฟล์

$length หมายถึง จำนวนตัวอักษรที่ต้องการบันทึกลงไฟล์ (มีหน่วยเป็นไบต์) หากไม่กำหนดจะบันทึกกลุ่มตัวอักษรทั้งหมดจากตัวแปร $string ลงในไฟล์

ตัวอย่างที่ 8.7 การใช้ฟังก์ชัน fputs ( )

```
1   <?php
2       $filename = "sitevisitors.txt";
3       if (file_exists ($filename)) {
4           $count = file($filename);
5           $count [0] ++;
6           $fp = fopen ($filename, "w");
7           fputs ($fp, "$count[0]");
8           fclose ($fp);
9           echo $count[0];
10      } else {
11          $fh = fopen ($filename, "w");
12          if ($fh==false) die ("ไม่สามารถสร้างไฟล์ได้");
13          fputs ($fh, 1);
14          fclose ($fh);
15          $count = file ($filename);
16          echo $count[0];
17      }
18  ?>
```

จากตัวอย่างที่ 8.7 การใช้ฟังก์ชัน fputs ( ) เพื่อพัฒนาระบบตัวนับจำนวนผู้เข้าชมเว็บไซต์ อธิบายดังนี้

บรรทัดที่ 2 กำหนดให้ตัวแปร $filename มีค่าเท่ากับ "sitevisitors.txt" (ชื่อของเท็กซ์ไฟล์)

บรรทัดที่ 3 ใช้โครงสร้างเงื่อนไข if ตรวจสอบ ผลการทำงานของฟังก์ชัน file_exit ( ) ใช้สำหรับตรวจสอบว่ามีไฟล์ที่ระบุหรือไม่ ถ้ามีไฟล์ที่ระบุจะคืนค่า true หากไม่มีไฟล์ที่ระบุจะคืนค่า false ในกรณีคืนค่า กรณีคืนค่า true ไปบรรทัดที่ 4 หากคืนค่า false ไปบรรทัด 10

บรรทัดที่ 4 กำหนดให้ตัวแปร $count อ่านไฟล์ข้อมูลจากไฟล์ "sitevisitors.txt" ด้วยฟังก์ชัน file ( ) รูปแบบการอ่านจะอ่านข้อมูลทั้งหมดออกมาในรูปแบบอาร์เรย์

บรรทัดที่ 5 กำหนดให้ตัวแปร $count [0] เพิ่มค่า 1 ค่า (การที่ต้องเรียกใช้ $count [0] เป็นเพราะผลมาจากฟังก์ชัน file ( ) เมื่ออ่านข้อมูลทั้งหมดตัวแปรจะเปลี่ยนเป็นชนิดอาร์เรย์ ดังนั้น การอ้างอิงค่าต้องระบุในรูปแบบของอาร์เรย์)

บรรทัดที่ 6 กำหนดให้ตัวแปร $fp ทำหน้าที่เป็นตัวชี้ตำแหน่งข้อมูลในไฟล์ เปิดไฟล์ "sitevisitors.txt" ในโหมด "w" ด้วยฟังก์ชัน fopen ( )

บรรทัดที่ 7 เขียนข้อมูลลงในไฟล์ ($fp คือ ตัวแทนของไฟล์ข้อมูล และ $count [0] คือ ข้อมูลที่จะใช้เขียนลงในไฟล์)

บรรทัดที่ 8 ปิดการเชื่อมต่อไฟล์

บรรทัดที่ 9 แสดงค่าผลในตัวแปร $count [0]

บรรทัดที่ 10 ปิดโครงสร้างเงื่อนไข if เริ่มต้นโครงสร้างเงื่อนไข else (กรณีเป็น false จากโครงสร้างเงื่อนไข if)

บรรทัดที่ 11 กำหนดให้ตัวแปร $fh ทำหน้าที่เป็นตัวชี้ตำแหน่งข้อมูลในไฟล์ เปิดไฟล์ "sitevisitors.txt" ในโหมด "w" ด้วยฟังก์ชัน fopen ( )

บรรทัดที่ 12 ตรวจสอบค่าตัวแปร $fh ด้วยโครงสร้างเงื่อนไข if ว่าสามารถเปิดไฟล์ได้หรือไม่ หากเป็น true ไปบรรทัดที่ 13 หากเป็น false ให้หยุดทำงานแล้วแจ้งข้อความ "ไม่สามารถสร้างไฟล์ได้"

บรรทัดที่ 13 เขียนค่าเลข 1 ลงใน $fh ($fh คือ ตัวแทนของไฟล์ข้อมูล)

บรรทัดที่ 14 ปิดการเชื่อมต่อไฟล์

บรรทัดที่ 15 กำหนดให้ตัวแปร $count อ่านไฟล์ข้อมูลจากไฟล์ "sitevisitors.txt" ด้วยฟังก์ชัน file ( ) รูปแบบการอ่านจะอ่านข้อมูลทั้งหมดออกมาในรูปแบบอาร์เรย์

บรรทัดที่ 16 แสดงค่าตัวแปร $count [0] (จะแสดงหมายเลข 1 ในรอบถัดๆ ไปจะทำงานเฉพาะช่วงบรรทัดที่ 3 ถึงบรรทัดที่ 9 เท่านั้น)

3) ฟังก์ชัน file_put_content ( ) เป็นฟังก์ชันสำหรับเขียนข้อมูลลงไฟล์ เช่นเดียวกับฟังก์ชัน fwrite ( ) และ fputs ( ) แต่ฟังก์ชันนี้จะดำเนินการเองตั้งแต่เปิดไฟล์ เขียนข้อมูลลงไฟล์ และปิดไฟล์ ต่างจากวิธีเดิมที่ต้องใช้ฟังก์ชัน fopen ( ) เปิดไฟล์ก่อน แล้วจึงใช้ฟังก์ชัน fwrite ( ) หรือฟังก์ชัน fputs ( ) เขียนข้อมูลลงไฟล์ เมื่อการทำงานเสร็จสิ้นจึงใช้ฟังก์ชัน fclose ( ) ปิดไฟล์ รูปแบบการใช้งานฟังก์ชัน file_put_content ( ) มีดังนี้

รูปแบบ

```
int file_put_contents ( string $filename , mixed $data )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่จะเขียนข้อมูล

$data หมายถึง ข้อมูลที่ต้องการบันทึกลงไฟล์

ตัวอย่างที่ 8.8 การใช้ฟังก์ชัน file_put_content ( )

```
<?php
    $file = 'people.txt';
    $current = file_get_contents ($file);          // เปิดไฟล์รับข้อมูลเดิม
    $current .= "John Smith \n";                   // เพิ่มข้อมูลบุคลากร
    file_put_contents ($file, $current);           // เขียนข้อมูลลงไฟล์
?>
```

8.2.4 การอ่านข้อมูลจากไฟล์

การอ่านข้อมูลจากไฟล์ มีขั้นตอนการทำงานเหมือนกับการเขียนไฟล์ คือ เปิดไฟล์ด้วยฟังก์ชัน fopen ( ) ก่อนจึงจะสามารถอ่านไฟล์ข้อมูลได้ เมื่อเลิกใช้ไฟล์แล้วจะต้องปิดไฟล์ด้วยฟังก์ชัน fclose ( ) สำหรับฟังก์ชันที่ใช้อ่านไฟล์ข้อมูลมีดังนี้

1) ฟังก์ชัน feof ( ) เป็นฟังก์ชันที่ใช้ตรวจสอบและอ่านข้อมูลในไฟล์ เมื่อข้อมูลทั้งหมดถูกอ่านฟังก์ชัน feof จะส่งคืนค่าเป็น true แต่ถ้ายังอ่านไม่หมด จะส่งคืนค่าเป็น false มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool feof ( resource $handle )
```

เมื่อ $handle หมายถึง ตัวแปรสำหรับชี้ข้อมูลในไฟล์

ตัวอย่างที่ 8.9 การใช้ฟังก์ชัน feof ( )

```
<?php
    $file = fopen ("data.txt", "r");
    while (!feof ($file)) {
        $char = fgetc ($file);
        echo $char . "<br>";
    }
    fclose($file);
?>
```

จากตัวอย่างที่ 8.9 การใช้ฟังก์ชัน feof ( ) เพื่ออ่านค่าข้อมูลในไฟล์ อธิบายบรรทัดที่สำคัญ ดังนี้

บรรทัดที่ 2 กำหนดให้ตัวแปร $file เท่ากับผลการเรียนใช้ฟังก์ชัน fopen ( ) เปิดไฟล์ "data.txt" ด้วยไฟล์โหมด "r"

บรรทัดที่ 3 ใช้โครงสร้างเงื่อนไขทำซ้ำ while เพื่ออ่านค่าข้อมูลตรวจสอบข้อมูลสุดท้ายด้วย feof (เพื่อข้อมูลไม่สามารถอ่านได้ฟังก์ชันจะคืนค่า false) โครงสร้างเงื่อนไขทำซ้ำ while จะวนรอบเพื่ออ่านข้อมูลไปเรื่อยๆ จนกว่าจะไม่มีข้อมูลให้อ่าน

2) ฟังก์ชัน fgetc ( ) ใช้อ่านข้อมูลจากไฟล์ครั้งละ 1 อักษร รูปแบบการใช้งานฟังก์ชันมีดังนี้

รูปแบบ

```
string fgetc ( resource $handle )
```

เมื่อ $handle หมายถึง ตัวแปรสำหรับชี้ข้อมูลในไฟล์

ตัวอย่างที่ 8.10 การใช้ฟังก์ชัน fgetc ( )

```
1   <?php
2       $fp = fopen ("somefile.txt", "r");
3       if (!$fp) {
4           echo "ไม่สามารถเปิดไฟล์ somefile.txt ได้";
5       } else {
6           while (false !== ($char = fgetc ($fp))) {
7               echo "$char<br>";
8           }
9       }
10  ?>
```

จากตัวอย่างที่ 8.10 แสดงตัวอย่างการใช้ฟังก์ชัน fgetc ( ) สำหรับอ่านข้อมูลจากไฟล์ครั้งละ 1 อักษร

3) ฟังก์ชัน fread ( ) ใช้อ่านข้อมูลจากไฟล์ตามความยาวของอักษรที่ระบุ รูปแบบการใช้งานฟังก์ชัน มีดังนี้

รูปแบบ

```
string fread ( resource $handle , int $length )
```

เมื่อ $handle หมายถึง ตัวแปรสำหรับชี้ตำแหน่งข้อมูลในไฟล์

$length หมายถึง ความยาวของอักษรที่ต้องการอ่าน

ตัวอย่างที่ 8.11 การใช้ฟังก์ชัน fread ( )

```
1   <?php
2       $filename = "/usr/local/something.txt";
3       $handle = fopen ($filename, "r");
4       $contents = fread ($handle, filesize ($filename));
```

```
5           fclose ($handle);
6       ?>
```

จากตัวอย่างที่ 8.11 แสดงตัวอย่างการใช้ฟังก์ชัน fread ( ) ใช้อ่านข้อมูลจากไฟล์ตามความยาวของอักษรที่ระบุ

4) ฟังก์ชัน fgets ( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูลจากไฟล์ครั้งละ 1 บรรทัด และจะอ่านข้อมูลจนกระทั่งพบอักษรสำหรับขึ้นบรรทัดใหม่ (\n) หรือสิ้นสุดไฟล์ จึงหยุดอ่าน มีรูปแบบการใช้งานฟังก์ชันมีดังนี้

รูปแบบ

```
string fgets ( resource $handle [, int $length ] )
```

เมื่อ $handle หมายถึง ตัวแปรสำหรับชี้ข้อมูลในไฟล์

$length หมายถึง ความยาวของอักษรที่ต้องการอ่าน

ตัวอย่างที่ 8.12 การใช้ฟังก์ชัน fgets ( )

```
1       <?php
2           $handle = @fopen ("/tmp/inputfile.txt", "r");
3           if ($handle) {
4               while (($buffer = fgets ($handle, 4096)) !== false) echo $buffer;
5               if (!feof ($handle)) {
6                   echo "Error: unexpected fgets ( ) fail\n";
7               }
8               fclose ($handle);
9           }
10      ?>
```

5) ฟังก์ชัน fgetss ( ) เป็นฟังก์ชันที่ให้ผลลัพธ์เหมือนกับฟังก์ชัน fgets ( ) แต่ต่างกันที่ฟังก์ชัน fgetss ( ) สามารถกำหนดให้แสดงหรือไม่แสดง HTML tags หรือ PHP ได้ โดยฟังก์ชัน fgets ( ) จะข้าม HTML tags หรือ PHP ไป แต่จะสนใจเฉพาะข้อความตัวอื่น ฟังก์ชัน fgetss ( ) มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
string fgetss ( resource $handle [, int $length [, string $allowable_tags ]] )
```

เมื่อ $handle หมายถึง ตัวแปรสำหรับชี้ข้อมูลในไฟล์

$length หมายถึง ความยาวของอักษรที่ต้องการอ่าน

$allowable_tags หมายถึง อนุญาตให้แท็กทำงานหรือไม่ทำงาน

ตัวอย่างที่ 8.13 การใช้ฟังก์ชัน fgetss ( )

```
<?php
    $str = <<<EOD
    <html><body>
    <p>Welcome! Today is the <?php echo (date ("jS")); ?> of <?= date ("F"); ?>.</p>
    </body></html>
    Text outside of the HTML block.
    EOD;
    file_put_contents ("sample.php", $str);
    $handle = @fopen ("sample.php", "r");
    if ($handle) {
        while (!feof ($handle)) {
            $buffer = fgetss ($handle, 4096);
            echo $buffer;
        }
        fclose ($handle);
    }
?>
```

6) ฟังก์ชัน readfile ( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูลในไฟล์ทั้งหมด สามารถเปิดไฟล์ได้เอง โดยไม่ต้องใช้ฟังก์ชัน fopen ( ) ฟังก์ชันจะแสดงข้อมูลที่อ่านได้ทางหน้าจอ พร้อมคืนค่าเป็นขนาดของข้อมูล (หน่วยเป็นไบต์) หากเกิดข้อผิดพลาด จะคืนค่าเป็น false รูปแบบการใช้งานฟังก์ชัน มีดังนี้

รูปแบบ

```
int readfile ( string $filename [, bool $use_include_path = false] )
```

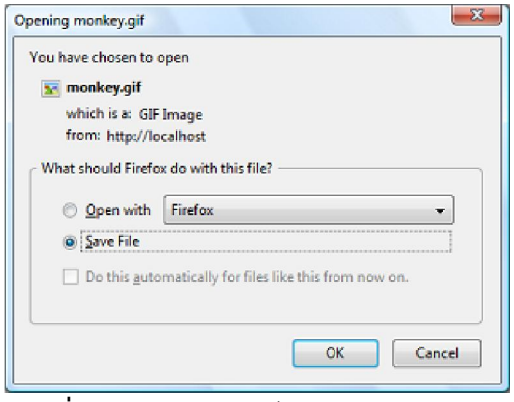
เมื่อ $filename หมายถึง ชื่อไฟล์ที่จะอ่านข้อมูล

$use_include_path หมายถึง กำหนดเป็น True หากต้องการให้ค้นหาไฟล์ใน include_path

ตัวอย่างที่ 8.14 การใช้ฟังก์ชัน readfile ( )

```
<?php
    $file = "monkey.gif";
    if (file_exists ($file)) {
        header ("Content-Description: File Transfer");
```

```
5               header ("Content-Type: application/octet-stream");
6               header ("Content-Disposition: attachment; filename=".basename ($file));
7               header ("Content-Transfer-Encoding: binary");
8               header ("Expires: 0");
9               header ("Cache-Control: must-revalidate");
10              header ("Pragma: public");
11              header ("Content-Length: " . filesize ($file));
12              ob_clean ( );  flush ( );
13              readfile ($file); exit;
14          }
15      ?>
```



ภาพที่ 8.1 แสดงการเปิดไฟล์รูปภาพ

7) ฟังก์ชัน fpassthru ( ) เป็นฟังก์ชันที่ใช้อ่านข้อมูลในไฟล์เหมือนกับฟังก์ชัน readfile ( ) แต่จะต้องเปิดไฟล์ด้วยฟังก์ชัน fopen ( ) ก่อน จึงจะใช้ฟังก์ชัน fpassthru ( ) อ่านข้อมูลได้ มีรูปแบบการใช้งานฟังก์ชัน ดังนี้

รูปแบบ

```
int fpassthru ( resource $handle )
```

เมื่อ $handle หมายถึง ตัวแปรสำหรับชี้ข้อมูลในไฟล์

ตัวอย่างที่ 8.15 การใช้ฟังก์ชัน fpassthru() สำหรับไฟล์ชนิดไบนารี

```
1       <?php
2           $name = "./img/ok.png";
3           $fp = fopen ($name, "rb");          // เปิดไฟล์ด้วยโหมดไบนารี
4           header ("Content-Type: image/png");
5           header ("Content-Length: " . filesize ($name));
```

```
6           fpassthru ($fp);
7           exit;
8       ?>
```

8) ฟังก์ชัน file ( ) เป็นฟังก์ชันสำหรับอ่านข้อมูลในไฟล์ทั้งหมดเหมือนกับฟังก์ชัน readfile ( ) แต่ต่างกันที่ฟังก์ชัน file ( ) จะนำข้อมูลเก็บลงอาร์เรย์ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
array file ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่ต้องการอ่านข้อมูล

ตัวอย่างที่ 8.16 การใช้ฟังก์ชัน file ( )

```
1       <?php
2           $lines = file ("http://www.example.com/");
3           foreach ($lines as $line_num => $line) {
4               echo "Line #<b>{$line_num}</b> : " . htmlspecialchars($line) . "<br>";
5           }
6           $html = implode ('', file ("http://www.example.com/"));
7           $trimmed = file("somefile.txt", FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
8       ?>
```

9) ฟังก์ชัน file_get_content ( ) เป็นฟังก์ชันสำหรับอ่านข้อมูลในไฟล์ทั้งหมดเหมือนกับฟังก์ชัน file ( ) แต่ต่างกันที่ฟังก์ชัน file_get_content ( ) จะคืนค่าเป็นข้อความ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
string file_get_contents ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่ต้องการอ่านข้อมูล

ตัวอย่างที่ 8.17 การใช้ฟังก์ชัน file_get_content ( ) หน้าโฮมเพจของเว็บไซต์

```
1       <?php
2           $homepage = file_get_contents ("http://www.example.com/");
3           echo $homepage;
4       ?>
```

ตัวอย่างที่ 8.18 การใช้ฟังก์ชัน file_get_content ( ) ค้นหาในตำแหน่ง include_path

```
1       <?php
2           $file = file_get_contents ("./people.txt", FILE_USE_INCLUDE_PATH);
3       ?>
```

8.2.5 การลบไฟล์

สามารถใช้ฟังก์ชัน unlink ( ) ได้ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool unlink ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่จะลบ

ตัวอย่างที่ 8.19 การใช้ฟังก์ชัน unlink ( )

```
1   <?php
2       $fh = fopen ("test.html", "a");
3       fwrite ($fh, "<h1>Hello world!</h1>");
4       fclose ($fh);
5       unlink ("test.html");
6   ?>
```

8.2.6 ฟังก์ชันอื่นๆ ที่ใช้จัดการกับไฟล์

1) ฟังก์ชันสำหรับการตรวจสอบไฟล์

จากตัวอย่างก่อนหน้านี้ หากไฟล์ที่ต้องการเปิดไม่มีอยู่จริง หรือไฟล์นั้นไม่อนุญาตให้มีการเขียน/อ่านไฟล์ จะทำให้ฟังก์ชันคืนค่าเป็นความผิดพลาด หากไม่ดักจับหรือตรวจสอบความผิดพลาดเหล่านั้นอาจทำให้มีผลกระทบต่อการทำงานของส่วนอื่นๆ สำหรับฟังก์ชันที่ใช้ตรวจสอบไฟล์ มีดังต่อไปนี้

ตารางที่ 8.3 ฟังก์ชันที่ใช้ตรวจสอบเกี่ยวกับไฟล์

ฟังก์ชัน	การทำงาน	รูปแบบ
is_file ( )	ตรวจสอบว่าเป็นไฟล์ทั่วไปหรือไม่ หากใช่จะคืนค่าเป็น True หากไม่ใช่ จะคืนค่าเป็น False	bool is_file (string $filename)
is_writable ( )	ตรวจสอบว่าสามารถเขียนข้อมูลลงในไฟล์ได้หรือไม่ หากได้จะคืนค่าเป็น True หากไม่ได้ จะคืนค่าเป็น False	bool is_writable (string $filename)
is_readable ( )	ตรวจสอบว่าสามารถอ่านข้อมูลจากไฟล์นั้นได้หรือไม่ การคืนค่าจะเหมือนกับฟังก์ชัน is_writable ( )	bool is_readable (string $filename)

2) การตรวจสอบเวลาที่เกี่ยวข้องกับไฟล์

ตารางที่ 8.4 ฟังก์ชันที่ใช้ตรวจสอบเวลาที่เกี่ยวข้องกับไฟล์

ฟังก์ชัน	การทำงาน	รูปแบบ
fileatime ( )	ใช้หาเวลา (ค่า Unix Timestamp) ที่มีการเข้าถึงไฟล์ครั้งล่าสุด	int fileatime (string $filename)
filectime ( )	ใช้หาเวลา (ค่า Unix Timestamp) ที่มีการเปลี่ยนแปลงข้อมูลของไฟล์ครั้งล่าสุด ข้อมูลของไฟล์ที่จะบันทึกได้แก่ การกำหนดกลุ่มผู้ใช้งาน มี 3 กลุ่ม ประกอบด้วย user, group และ ownership การกำหนดสิทธิ์การเข้าถึงไฟล์ โดยเวลานี้จะถูกบันทึกขณะสร้างไฟล์และทุกครั้งที่มีการเปลี่ยนแปลงข้อมูล	int filectime (string $filename)
filemtime ( )	ใช้หาเวลา (ค่า Unix Timestamp) ที่มีการแก้ไขข้อมูลในไฟล์ครั้งล่าสุด โดยเวลานี้จะถูกบันทึกตอนสร้างไฟล์ และทุกครั้งที่มีการแก้ไขข้อมูล	int filemtime (string $filename)

8.3 การจัดการกับไดเรกทอรี

ภาษา PHP มีฟังก์ชันที่ใช้จัดการเกี่ยวกับไดเรกทอรี ดังนี้

8.3.1 การอ่านไดเรกทอรี

1) ฟังก์ชัน opendir ( ) เป็นฟังก์ชันสำหรับเปิดไดเรกทอรีเพื่ออ่านรายชื่อไฟล์ภายใน โดยจะส่งคืนค่าเป็นตัวชี้ตำแหน่งไดเรกทอรี (Directory Hander) หากเปิดไดเรกทอรีไม่ได้ จะคืนค่าเป็น False มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
resource opendir ( string $path )
```

เมื่อ $path หมายถึง ไดเรกทอรีที่ต้องการเปิด

2) ฟังก์ชัน closedir ( ) เป็นฟังก์ชันสำหรับปิดไดเรกทอรี มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
void closedir ([ resource $dir_handle ] )
```

เมื่อ $dir_handle หมายถึง ตัวแปรที่ใช้เก็บค่าของตัวชี้ที่ใช้ชี้ไดเรกทอรี

3) ฟังก์ชัน readdir ( ) เป็นฟังก์ชันสำหรับอ่านไดเรกทอรี มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
string readdir ([ resource $dir_handle ] )
```

เมื่อ $dir_handle หมายถึง ตัวแปรที่ใช้เก็บค่าของตัวชี้ที่ใช้ชี้ไดเรกทอรี

ตัวอย่างที่ 8.20 การใช้ฟังก์ชัน opendir ( ), closedir ( ) และ readdir ( )

```
1   <?php
2       $dir = "/etc/php5/";
3       if (is_dir ($dir)) {
4           if ($dh = opendir ($dir)) {
5               while (($file = readdir ($dh)) !== false) {
6                   echo "filename: $file : filetype: " . filetype ($dir . $file) . "<br>";
7               }
8               closedir ($dh);
9           }
10      }
11  ?>
```

4) ฟังก์ชัน rewinddir ( ) เป็นฟังก์ชันสำหรับกลับไปที่ส่วนต้นของไดเรกทอรี มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
void rewinddir ([ resource $dir_handle ] )
```

เมื่อ $dir_handle หมายถึง ตัวแปรที่ใช้เก็บค่าของตัวชี้ที่ใช้ชี้ไดเรกทอรี

8.3.2 การดูรายละเอียดของไดเรกทอรี

1) ฟังก์ชัน dirname ( ) เป็นฟังก์ชันที่ใช้สำหรับแสดงพาธของไดเรกทอรี มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
string dirname ( string $path )
```

เมื่อ $path หมายถึง พาธของไดเรกทอรี

2) ฟังก์ชัน basename ( ) เป็นฟังก์ชันที่ใช้สำหรับแสดงชื่อของไฟล์ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
string basename ( string $path [, string $suffix ] )
```

เมื่อ $path หมายถึง พาธของไดเรกทอรี

$suffix หมายถึง ส่วนขยายไฟล์นั้น เช่น .php, .html, .txt หากไม่กำหนดค่าที่ได้จะมีทั้งชื่อไฟล์ และส่วนขยาย

ตัวอย่างที่ 8.21 การใช้ฟังก์ชัน basename ( )

```
1   <?php
2       $p = “c:/windows/php.ini” ;
3       $basel1 = basename ($p);              // ผลลัพธ์ คือ $base1 =” php.ini”
4       $basel2 = basename ($p, “ .ini”);     // ผลลัพธ์ คือ $base2 = “php”
5   ?>
```

2) ฟังก์ชัน realpath ( ) เป็นฟังก์ชันที่ใช้สำหรับแสดงพาธจริงทั้งหมด มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
string realpath ( string $path )
```

เมื่อ $path หมายถึง พาธของไดเรกทอรี

ตัวอย่างที่ 8.22 การใช้ฟังก์ชัน realpath ( )

```
<?php
    echo realpath('/windows/system32');          // ผลลัพธ์ คือ C:\WINDOWS\System32
?>
```

3) ฟังก์ชัน pathinfo ( ) เป็นฟังก์ชันที่ใช้สำหรับตรวจสอบพาธโดยการส่งคืนค่ากลับมาเป็นอาร์เรย์ที่สมาชิกแต่ละตัวประกอบไปด้วย ชื่อไดเรกทอรี, basename และส่วนขยายของไฟล์ อาร์เรย์ผลลัพธ์ที่จะได้อยู่ในรูปแบบคีย์และค่าอาร์เรย์ โดยค่าคีย์อาร์เรย์ จะประกอบไปด้วย dirname , basename และ extension มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
mixed pathinfo ( string $path [, int $options = PATHINFO_DIRNAME | PATHINFO_BASENAME | PATHINFO_EXTENSION | PATHINFO_FILENAME ] )
```

เมื่อ $path หมายถึง พาธของไดเรกทอรี

$options หมายถึง กำหนดเพื่อดึงเฉพาะข้อมูลที่ต้องการ มี 4 ค่า ได้แก่

PATHINFO_DIRNAME เฉพาะข้อมูลไดเรกทอรี

PATHINFO_BASENAME เฉพาะข้อมูลชื่อไฟล์และนามสกุลไฟล์

PATHINFO_EXTENSION เฉพาะข้อมูลนามสกุลของไฟล์

PATHINFO_FILENAME เฉพาะข้อมูลชื่อไฟล์

หมายเหตุ

หากไม่ระบุพารามิเตอร์ฟังก์ชัน pathinfo ( ) จะคืนค่าเป็นอาร์เรย์ประกอบด้วยอินเด็กซ์ dirname, basename, extention, และ filename ใช้สำหรับเก็บค่าไดเรกทอรี ชื่อไฟล์ (รวมนามสกุลไฟล์) และชื่อไฟล์ ตามลำดับ

ตัวอย่างที่ 8.23 การใช้ฟังก์ชัน pathinfo ( )

```
1   <?php
2       $path_parts = pathinfo ("/www/htdocs/inc/lib.inc.php ");
3       echo $path_parts ["dirname "]. "<br>";
4       echo $path_parts ["basename "] . "<br>";
5       echo $path_parts ["extension "] . "<br>"";
6       echo $path_parts ["filename "] . "<br>";
7   ?>
```

ผลลัพธ์

```
/www/htdocs/inc
lib.inc.php
php
lib.inc
```

8.3.3 การสร้างและลบไดเรกทอรี

1) ฟังก์ชัน mkdir ( ) เป็นฟังก์ชันที่ใช้สำหรับสร้างไดเรกทอรี หากสร้างได้ ฟังก์ชันจะส่งคืนค่าเป็น true หากสร้างไม่ได้ เพราะมีไดเรกทอรีนั้นอยู่แล้ว หรือไม่ได้รับสิทธิ์ให้สร้างไดเรกทอรี (ขึ้นอยู่กับระบบความปลอดภัยของแต่ละเครื่อง และการกำหนดค่าโหมดของไฟล์) ฟังก์ชันจะส่งคืนค่าเป็น false มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool mkdir ( string $pathname [, int $mode = 0777 ] )
```

เมื่อ $pathname คือ พาธของไดเรกทอรี

$mode คือ โหมดของไดเรกทอรีที่ใช้ใน Unix เท่านั้น ในระบบปฏิบัติการ MS-Windows พารามิเตอร์ตัวนี้จะไม่ถูกนำมาใช้ โดยปริยายกำหนดให้มีค่าโหมดเป็น 0777 สามารถศึกษารายละเอียดเพิ่มเติมโหมดในฟังก์ชัน chmod ( )

ตัวอย่างที่ 8.24 การใช้ฟังก์ชัน mkdir ( )

```
1    <?php
2        mkdir ("/path/to/my/dir", 0700);  // สร้างไดเรกทอรีพร้อมกำหนดโหมดเป็น 0700
3    ?>
```

2) ฟังก์ชัน rmdir ( ) เป็นฟังก์ชันที่ใช้สำหรับลบไดเรกทอรี มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool rmdir ( string $dirname )
```

เมื่อ $dirname หมายถึง พาธของไดเรกทอรีที่ต้องการลบ

ตัวอย่างที่ 8.25 การใช้ฟังก์ชัน rmkdir ( )

```
1    <?php
2        if (!is_dir ("examples ")) {
3            mkdir ("examples ");
4        }
5        rmdir ("examples ");
6    ?>
```

8.3.4 การตรวจสอบการมีอยู่ของไฟล์และไดเรกทอรี

ฟังก์ชัน file_exists ( ) เป็นฟังก์ชันที่ใช้สำหรับตรวจสอบว่ามีไฟล์ หรือไดเรกทอรีตามที่ระบุอยู่หรือไม่ หากมีจะคืนค่ากลับมาเป็นค่า true

รูปแบบ

```
bool file_exists ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อของไฟล์หรือไดเรกทอรีที่ต้องการตรวจสอบ

ตัวอย่างที่ 8.26 การใช้ฟังก์ชัน file_exists ( )

```
1    <?php
2        $filename = "/path/to/foo.txt";
3        if (file_exists ($filename)) {
4            echo "ตรวจพบไฟล์เอกสาร $filename";
5        } else {
6            echo "ไม่พบไฟล์เอกสาร $filename";
7        }
8    ?>
```

8.3.5 การคัดลอกไฟล์ เปลี่ยนแปลงชื่อไฟล์ และย้ายไดเรกทอรี

ฟังก์ชัน copy ( ) เป็นฟังก์ชันที่ใช้สำหรับคัดลอกไฟล์ โดยกำหนดชื่อไฟล์ต้นฉบับ และชื่อไฟล์ปลายทางที่นำไฟล์คัดลอกไปไว้ หากกำหนดไดเรกทอรีรวมกับชื่อไฟล์ จะหมายความว่า ให้คัดลอกไฟล์ไปยังไดเรกทอรีที่ระบุ หากไดเรกทอรีของปลายทางมีชื่อไฟล์ซ้ำกัน ฟังก์ชันจะนำไฟล์ที่คัดลอกทับไฟล์เดิมที่มีอยู่ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool copy ( string $source , string $dest )
```

เมื่อ $source หมายถึง ชื่อไฟล์ต้นฉบับ

$dest หมายถึง ชื่อไฟล์หรือตำแหน่งปลายทางที่จะคัดลอกไฟล์

ตัวอย่างที่ 8.27 การใช้ฟังก์ชัน copy ( )

```
<?php
    $file = 'example.txt';
    $newfile = 'example.txt.bak';
    if  (!copy ($file, $newfile)) {
        echo "ไม่สามารถคัดลอกไฟล์ $file... ได้ <br>";
    }
?>
```

8.3.6 การเปลี่ยนชื่อไฟล์และย้ายไดเรกทอรี

ฟังก์ชัน rename ( ) เป็นฟังก์ชันที่ใช้สำหรับเปลี่ยนชื่อไฟล์ หากมีการกำหนดไดเรกทอรีร่วมกับชื่อไฟล์ใหม่ จะหมายถึงให้ย้ายไฟล์ไปยังไดเรกทอรีที่กำหนดด้วย และหากไดเรกทอรีนั้นมีชื่อไฟล์ซ้ำกัน ฟังก์ชันจะเขียนไฟล์ใหม่ทับไฟล์เดิมที่มีอยู่ มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool rename ( string $oldname , string $newname )
```

เมื่อ $oldname หมายถึง ชื่อไฟล์เดิม

$newname หมายถึง ชื่อไฟล์ใหม่

ตัวอย่างที่ 8.28 การใช้ฟังก์ชัน rename ( )

```
1   <?php
2       rename ("/tmp/tmp_file.txt", "/home/user/login/docs/my_file.txt");
3   ?>
```

8.4 การอัพโหลดไฟล์

ตัวแปร $_FILES เป็นตัวแปรประเภทตัวแปรพิเศษ ที่โปรแกรมภาษา PHP กำหนดไว้เพื่อเข้าถึงการทำงานได้ทุกส่วนของสคริปต์ โดยตัวแปร $_FILES จะเก็บข้อมูลที่เกี่ยวกับไฟล์อัพโหลด (จะต้องใช้รูปแบบการส่งฟอร์มแบบ HTTP POST method เท่านั้น) มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
$_FILES [userfile] [key]
```

เมื่อ userfile หมายถึง ชื่อที่กำหนดในแอทริบิวต์ name ที่ใช้รับชื่อไฟล์ที่ต้องการอัพโหลดจากเว็บฟอร์ม

key หมายถึง คีย์ในตัวแปร $_FILES ที่ใช้เก็บค่าต่างๆ ของไฟล์ โดยที่ค่าคีย์ของตัวแปร $_FILES มีดังนี้

ตารางที่ 8.5 ค่าคีย์ของตัวแปร $_FILES

คีย์	คำอธิบาย
name	ชื่อของไฟล์อัพโหลด
type	ชนิดของไฟล์อัพโหลด
tmp_name	ไดเรกทอรีฝั่งเซิร์ฟเวอร์สำหรับเก็บไฟล์อัพโหลดชั่วคราว ซึ่งพาธก็คือค่าของ "upload_tmp_dir" ส่วนชื่อไฟล์ PHP จะสร้างให้โดยอัตโนมัติ
error	ข้อผิดพลาดที่เกิดจากการอัพโหลด จะส่งคืนค่าแสดงเป็นตัวเลข มี 4 ค่า ได้แก่ ■ 0 หมายถึง การอัพโหลดสมบูรณ์ ■ 1 หมายถึง ขนาดไฟล์ใหญ่กว่าที่กำหนดใน upload_max_filesize (กำหนดใน php.ini) ■ 2 หมายถึง ขนาดไฟล์ใหญ่กว่าที่กำหนดใน MAX_FILE_SIZE ซึ่งกำหนดในฟอร์ม ■ 3 หมายถึง มีเพียงเฉพาะบางส่วนของไฟล์เท่านั้นที่ถูกอัพโหลด ■ 4 หมายถึง ไม่มีไฟล์ที่อัพโหลด
size	ขนาดของไฟล์อัพโหลด (หน่วยเป็นไบต์)

8.4.1 ฟังก์ชันเกี่ยวกับการอัพโหลด

1) ฟังก์ชัน is_upload_file ( ) เป็นฟังก์ชันที่ใช้สำหรับตรวจสอบว่ามีการส่งไฟล์ที่ต้องการอัพโหลดหรือไม่ และไฟล์ที่ส่งมาเป็นแบบ POST หรือไม่ การอัพโหลดไฟล์จากเว็บฟอร์มจะต้องส่งแบบ POST เท่านั้น เพราะสามารถส่งข้อมูลที่มีความยาวไม่จำกัด (การส่งแบบ GET จะจำกัดความยาวในการส่งได้ไม่เกิน 255 ตัวอักษร) มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool is_uploaded_file ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่ต้องการตรวจสอบ

ตัวอย่างที่ 8.29 การใช้ฟังก์ชัน is_upload_file ( )

```
<?php
    if (is_uploaded_file ($_FILES ['userfile'] ['tmp_name'])) {
        echo "ไฟล์ ". $_FILES ['userfile'] ['name'] ." อัพโหลดสำเร็จแล้ว<br>";
        echo "มีรายละเอียด ดังต่อไปนี้ <br>";
        readfile ($_FILES ['userfile'] ['tmp_name']);
    } else {
        echo "การอัพโหลดไม่สำเร็จ กรุณาตรวจสอบไฟล์ใหม่ ::";
        echo "ชื่อไฟล์ คือ '". $_FILES ['userfile'] ['tmp_name'] . "'.";
    }
?>
```

2) ฟังก์ชัน move_upload_file ( ) เนื่องจากการอัพโหลดไฟล์ของ PHP จะเก็บในไดเรกทอรีชั่วคราวก่อน แล้วจึงใช้ฟังก์ชัน move_upload_file ( ) ย้ายไฟล์ที่ได้ไปเก็บในไดเรกทอรีปลายทางที่กำหนด หากย้ายไฟล์ได้สำเร็จ จะคืนค่าเป็น true หากไม่สำเร็จ จะคืนค่าเป็น false มีรูปแบบการใช้งานฟังก์ชันดังนี้

รูปแบบ

```
bool move_uploaded_file ( string $filename , string $destination )
```

เมื่อ $filename หมายถึง ชื่อไฟล์ที่ต้องการอัพโหลด

$destination หมายถึง ตำแหน่งไดเรกทอรีปลายทางที่จะเก็บไฟล์อัพโหลด

ตัวอย่างที่ 8.30 ฟอร์มอัพโหลดไฟล์ (แบบครั้งละ 1 ไฟล์)

```
//ก่อนอื่นจะส่งไฟล์เข้าเครื่องเซิร์ฟเวอร์ต้องแก้ฟอร์มโดยเพิ่ม enctype="multipart/form-data"
//และส่งไฟล์โดยใช้อินพุทชนิดไฟล์
<form action="uploading.php" method="post" enctype="multipart/form-data">
<br /><input type="file" name="picture" />
</form>
<?php  // การใช้ฟังก์ชัน move_upload_file ( ) เพื่ออัพโหลดไฟล์
    if (isset ($_FILES ['picture'])) { // ตรวจดูก่อนว่าอัพโหลดไฟล์เข้ามาจริงๆ
        move_uploaded_file ($_FILES ['picture'] ['name']
        ,'uploads/' . $_FILES ['picture'] ['tmp_name']);
    }
?>
```

ตัวอย่างที่ 8.31 ฟอร์มอัพโหลดไฟล์ (แบบหลายไฟล์)

```
//ก่อนอื่นจะส่งไฟล์เข้าเครื่องเซิร์ฟเวอร์ต้องแก้ฟอร์มโดยเพิ่ม enctype="multipart/form-data"
//และส่งไฟล์โดยใช้อินพุทชนิดไฟล์
<form action="uploading.php" method="post" enctype="multipart/form-data">
<br /><input type="file" name="pictures [ ]" />
<br /><input type="file" name="pictures [ ]" />
<br /><input type="file" name="pictures [ ]" />
<br /><input type="file" name="pictures [ ]" />
<br /><input type="file" name="pictures [ ]" />
<input type="submit" />
</form>
<?php  // move_upload_file ( ) เพื่ออัพโหลดไฟล์
        if ( isset($_FILES ['pictures'])) {
                foreach ($_FILES ['pictures'] ['error'] as $index => $value) {
                        if ($value == UPLOAD_ERR_OK) {
                                move_uploaded_file($_FILES['pictures']['tmp_name'][$index]
,'uploads/'.$_FILES['pictures']['name'][$index]);
                        }
                }
        }
?>
```

8.4.2 ฟังก์ชันการหาขนาด และชนิดของไฟล์

1) ฟังก์ชัน filesize ( ) เป็นฟังก์ชันที่ใช้สำหรับหาขนาดของไฟล์ ค่าที่ได้จะมีหน่วยเป็น byte หากต้องการเปลี่ยนหน่วยเป็น KB ต้องหารด้วย 1,024 หากต้องการเปลี่ยนเป็น MB ต้องหารด้วย 1,048,576

รูปแบบ

```
int filesize ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อของไฟล์ที่ต้องการหาขนาดของไฟล์

ตัวอย่างที่ 8.32 การใช้ฟังก์ชัน filesize ( )

```
1       <?php
2           $filename = 'somefile.txt';
```

```
3           echo $filename . ': ' . filesize($filename) . ' bytes';
4       ?>
```

2) ฟังก์ชัน filetype ( ) เป็นฟังก์ชันที่ใช้สำหรับตรวจสอบว่าไฟล์หรือไดเร็กทอรีที่ระบุ

- หากเป็นไฟล์ (File) จะคืนค่ากลับมาเป็นคำว่า file
- หากเป็นไดเร็กทอรี (Directory) จะคืนค่าเป็นคำว่า dir

นอกจากนี้แล้วอาจเป็นไปได้อื่นๆ เช่น char, block, link, unknown

รูปแบบ

```
string filetype ( string $filename )
```

เมื่อ $filename หมายถึง ชื่อของไฟล์ที่ต้องการหาขนาดของไฟล์

ตัวอย่างที่ 8.33 การใช้ฟังก์ชัน filetype ( )

```
1       <?php
2           echo filetype ("/etc/passwd");          // file
3           echo filetype ("/etc/");                // dir
4       ?>
```

สรุป

ในบทนี้ได้กล่าวถึงการจัดการกับไฟล์และไดเรกทอรี การประยุกต์ใช้ฟังก์ชันสำหรับจัดการกับระบบไฟล์และไดเรกทอรี ตัวอย่างเช่น การอ้างอิงตำแหน่งไฟล์และไดเรกทอรี การเปิดและปิดไฟล์ การเขียนและอ่านไฟล์ การจัดการตัวชี้ตำแหน่งเพื่ออ่านและเขียนค่าลงในไฟล์ การลบไฟล์ การสร้างไดเรกทอรี การลบไดเรกทอรี การเปลี่ยนชื่อไดเรอทอรี การย้ายตำแหน่ง และการอัพโหลดไฟล์ เป็นต้น กิจกรรมทั้งหมดที่กล่าวถึงจะต้องอยู่ภายใต้กรอบและสิทธิในการเข้าถึงไฟล์และไดเรกทอรี ตามแนวนโยบายที่ผู้ดูแลระบบเครือข่ายหรือผู้ดูแลเว็บเซิร์ฟเวอร์เป็นผู้กำหนด เพื่อความปลอดภัยของไฟล์ข้อมูลในระบบ ดังนั้นบางฟังก์ชันอาจจะใช้งานไม่ได้บนเว็บเซิร์ฟเวอร์หากไม่มีสิทธิในการเข้าถึง

คำถามท้ายบท

1. จงอธิบายความแตกต่างระหว่างการอ้างอิงตำแหน่งของไฟล์และไดเรกทอรีแบบ Absolute Path และการอ้างอิงแบบ Relative Path

2. จงอธิบายรายละเอียดของฟังก์ชัน fopen ( ) โหมดที่เกี่ยวข้อง และยกตัวอย่างการเขียนสคริปต์เพื่อเปิดไฟล์สำหรับการเขียนและอ่าน โดยเริ่มทำงานจากจุดสิ้นสุดของข้อมูลที่มีอยู่ ถ้าไม่มีไฟล์จะทำการสร้างไฟล์ขึ้นมาใหม่

3. จงอธิบายความหมายและหน้าที่ของตัวชี้ตำแหน่งข้อมูลในไฟล์

4. จงอธิบายหลักการทำงานของสคริปต์ด้านล่างตั้งแต่บรรทัดที่ 1 ถึงบรรทัดสุดท้าย และบอกผลลัพธ์ที่ได้จากสคริปต์

```
1   <?php
2       if (file_exists "file.txt"))) {
3           $reading = file ("file.txt");
4           $reading [0] ++;
5           $writing = fopen ("file.txt", "w");
6           fputs ($file_pointer, $reading [0]);
7           fclose ($writing);
8           echo $reading [0];
9       } else {
10          $writing = fopen ("file.txt", "w");
11          if ($writing==false) die ("ไม่สามารถสร้างไฟล์ได้");
12          fputs ($writing, 1);
13          fclose ($writing);
14          $reading = file ("file.txt");
15          echo $reading [0];
16      }
17  ?>
```

5. จงบอกชื่อและอธิบายหน้าที่ของฟังก์ชันในภาษา PHP ที่ใช้สำหรับบริหารจัดการไดเรกทอรี