**บทที่ 2**

**ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง**

ในบทนี้จะกล่าวถึงทฤษฎีและเทคโนโลยีที่เกี่ยวข้องกับการพัฒนาการอัพเกรดระบบจำลองการแลกเปลี่ยนเงินตราระหว่างประเทศ โดยผู้จัดทำได้ศึกษาทฤษฎีต่างๆ และเทคโนโลยีที่เกี่ยวข้องกับการทำวิจัยครั้งนี้ โดยสามารถแบ่งเป็นหัวข้อต่างๆ ได้ดังนี้

2.1 การบูรณาการอย่างต่อเนื่อง (Continuous Integration)

2.2 เครื่องมือเจนกิ้นส์ (Jenkins)

2.3 เชลล์ (Shell)

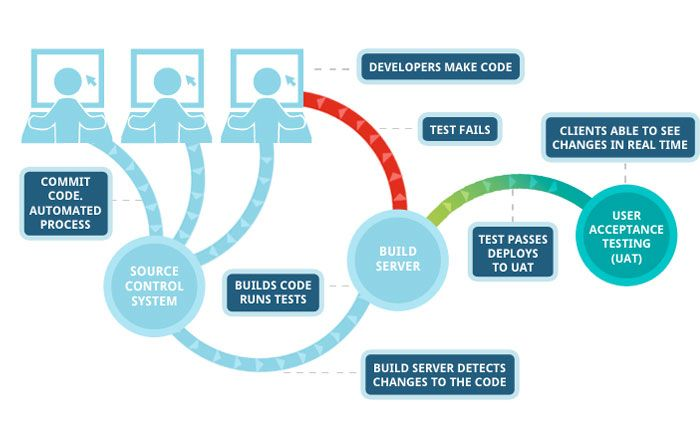
2.4 ระบบควบคุมรุ่น (Version Control)

**2.1 การบูรณาการอย่างต่อเนื่อง (Continuous Integration)**

**2.1.1 ความหมายของการบูรณาการอย่างต่อเนื่อง**

การบูรณาการอย่างต่อเนื่อง ชื่อภาษาอังกฤษ คือ Continuous Integration หมายถึง หลักการปฏิบัติของกาารทำงานเป็นทีม โดยเป็นส่วนหนึ่งของวิศวกรรมซอฟต์แวร์ มีการพัฒนาซอฟต์แวร์เป็นส่วนหนึ่งของเอจาย (Agile)  ก่อตั้งโดย Grady Booch ในปีคริตส์ศักราชที่ 1991 ถึงแแม้ว่า Booch จะไม่สนับสนุนการอินทิเกรชัน (Integration หมายถึงการ เอาโค้ดที่ผู้พัฒนา (Developer) เข้าสู่ซอสโค้ด (Source code) ส่วนกลางอย่างต่อเนื่อง เพื่อที่ระบบกลางจะสร้างแพ็กเกจให้ลูกค้าได้อย่างต่อเนื่อง) หลายๆ ครั้งต่อวัน การบูรณาการอย่างต่อเนื่องนั้นเป็นส่วนที่สนับสนุนเอ็กซ์ตรีมโปรแกรมมิ่ง (Extreme Programming) คือสามารถให้อินทิเกรชันได้มากกว่า 1 ครั้งต่อวัน หรืออาจจะถึง 10 ครั้งต่อวัน

เป้าหมายของ Continuous Integration นั้นคือ การป้องกันปัญหาของการส่งมอบโค้ดไปยังส่วนกลาง และยังช่วยในเรื่องการปรับปรุงคุณภาพของซอฟต์แวร์ ลดการใช้เวลาในการส่งมอบซอฟต์แวร์ให้กับลูกค้า การบูรณาการอย่างต่อเนื่องนั้นเกิดขึ้นได้เพราะได้รับแรงบันดาลในมากจากหลักการเอ็กซ์ตรีมโปรแกรมมิ่ง กล่าวได้ว่า เมื่อทางทีมผู้พัฒนาได้มีการส่งมอบโค้ดไปยังส่วนกลางมากขึ้นจนทำให้ซอสโค้ดส่วนกลางมีการเปลี่ยนแปลงมากจากของเดิมมาก อาจจะทำให้เกิดอินทิเกรชั่น เฮล (Integration hell) ได้ และอาจจะเกิดการขัดแย้งกันในการทำงานของซอฟต์แวร์ ทำให้ทางผู้พัฒนาต้องแก้ไขปัญหา และอาจจะใช้เวลานานในการแก้ไขปัญหา ดังนั้นหลังการบูรณาการอย่างต่อเนื่องนั้นมาช่วยลดการเกิดอินทีเกรชัน เฮล

[](http://www.glurgeek.com/education/trick-%E0%B8%AB%E0%B8%A5%E0%B8%B1%E0%B8%81%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2-ci/)

**รูปที่ 2.1** แสดงการทำงานของ Continuous Integration

ในการพัฒนานั้น จะเริ่มกันตั้งแต่การอินทิเกรชัน กันเมื่อมีการเปลี่ยนแปลงซอสโค้ดที่ Repository กลาง ระบบจะทำการตรวจสอบโค้ด หลังจากการเปลี่ยนแปลงว่าทำงานร่วมกันได้หรือไม่ ตั้งแต่การคอมไพล์ (Compile) จนไปถึงการทดสอบ (Testing) สิ่งที่สำคัญในการทำงานคือ ต้องได้ผลการทำงาน (Feedback) ที่รวดเร็วในการอินทิเกรชัน เพราะจะทำให้มองเห็นปัญหาที่เกิดขึ้นของระบบ ทำให้ช่วยลดเวลาในการค้นหาปัญหาและที่สำคัญช่วยลดเวลาของภาพรวมของระบบการพัฒนาได้ด้วย

**2.1.2 ขั้นตอนการทำงานของแนวคิด Continuous Integration**

1. ผู้พัฒนาแต่ละคนจะต้องสามารถ Build\* ทดสอบ และ Run\*\* ระบบที่เครื่องของตัวเองได้
2. เมื่อทำการพัฒนาฟีเจอร์ (Feature) เสร็จ ต้องมีการทดสอบอัตโนมัติเสมอ
3. ทำการดึงซอสโค้ดล่าสุดจาก Repository ของระบบ เพื่อตรวจสอบว่ามีกสรเปลี่ยนแปลงหรือไม่

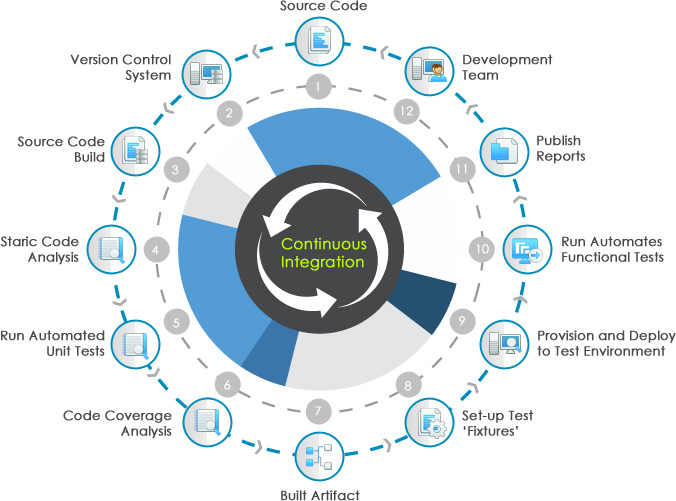
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*Build หมายถึง การที่เอาชุดคำสั่งของโปรแกรมและทรัพยากรต่างๆ ที่เขียนมาคอมไพล์ (Compile) และมาห่อรวมกัน (package) ให้ได้ผลลัพธ์สุดท้าย

\*\*Run หมายถึง การทำงานของเครื่องคอมพิวเตอร์ที่เป็นไปตามชุดคำสั่ง

* 1. ถ้ามีการเปลี่ยนแปลงให้ทำการผสาน (Merge) ที่เครื่องของผู้พัฒนาก่อน จากนั้นจึง build, ทดสอบ และ run อีกรอบ

1. หากเกิดการขัดแย้งหรือปัญหาให้แก้ไขที่เครื่องของผู้พัฒนาก่อนเสมอ
2. เมื่อทำการทดสอบทุกอย่างผ่านหมดในเครื่องของผู้พัฒนา ให้ส่งการเปลี่ยนแปลงไปที่ Repository กลาง
3. เมื่อซอสโค้ด (Repository) กลาง มีการเปลี่ยนแปลง ระบบจะทำการ Build, ทดสอบ และ run อีกรอบ
4. ถ้าหากเกิดความผิดพลาด ระบบจะส่งการแจ้งเตือนไปยังทีมที่พัฒนาทันที่ เพื่อให้เกิดการแก้ไข
5. เมื่อระบบทำงานเรียบร้อยจะส่งผลลัพธ์ไปยังส่วนการทำงานอื่นต่อไป เช่นการทดสอบระบบ การติดตั้งระบบ เป็นต้น

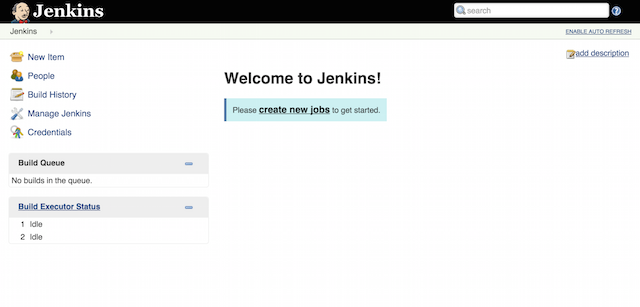


**รูปที่ 2.2** รูปแสดงขั้นตอนการทำงานของ Continuous Integration

**2.2 เครื่องมือเจนกิ้นส์ (Jenkins)**

**2.2.1 ความหมายของเจนกิ้นส์**

**เจนกิ้นส์ ( Jenkins)**  คือ เครื่องมือของ Continuous Integraion  ที่เขียนโดยภาษาจาวา (JAVA) เป็นโอเพนซอร์ส (Open Source) สำหรับการทำงานโดยอัตโนมัติของการบิ้วเซิฟเวอร์ (Build Server) ทำให้การพัฒนาการซอฟต์แวร์ทำได้เร็วมากยิ่งขึ้น เจนกิ้นส์เป็นเครื่องเมือที่ใช้จัดการและคุบควมวัฏจักรการกระบวนการการพัฒนาทั้งหมด ตั้งแต่การbuild สร้างเอกสาร การทดสอบ การส่งมอบแพ็กเกจ เป็นต้น



**รูปที่ 2.3** หน้าต่างของเครื่องมือเจนกิ้นส์

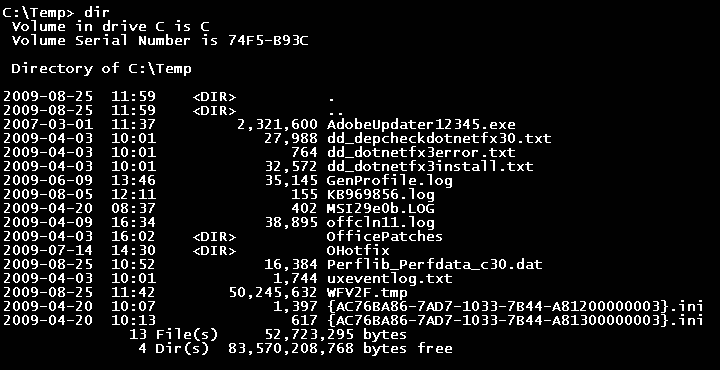
**2.3 เชลล์ (Shell)**

**2.3.1 ความหมายของเชลล์**

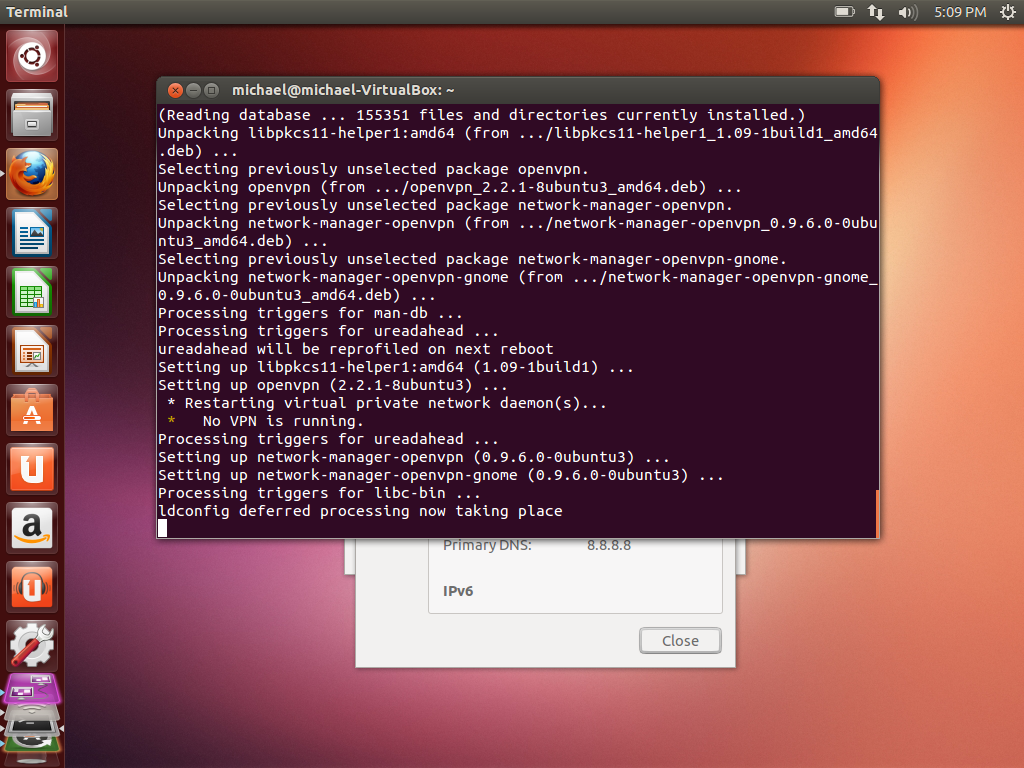
เชลล์ (Shell)คือ ส่วนประสานงานโต้ตอบระหว่างผู้ใช้กับระบบปฏิบัติการ (Operating System) ทำให้ที่แปลงคำสั่งจากผู้ใช้ให้ระบบปฏิบัติการเข้าใจ โปรแกรมเชลล์นั้นยังมีคุณสมบัติของภาษาของโปรแกรมอีกด้วย ดังนั้นทำให้สามารถใช้งานในรูปแบบของโปรแกรมโดยเอารูปแบบของคำสั่งมารวมกัน เรียกว่าเชลล์สคริปต์ (Shell script) ซึ่งเชลล์สามารถแบ่งได้เป็น 2 ประเภท คือ

1. ประเภทชุดคำสั่ง (Command line interface, CLI)

เป็นส่วนประสานงานกับผู้ใช้ที่อนุญาตให้ป้อนรูปแบบคำสั่งที่เป็นตัวหนังสือสั่งการลงไปด้วยตนเองเพื่อให้คอมพิวเตอร์ทำงานตามที่ต้องการทีละบรรทัดคำสั่ง



**รูปที่ 2.4** หน้าต่างโปรแกรม Command Prompt ในวินโดว์

****

**รูปที่ 2.5** หน้าต่างโปรแกรม Terminal ของยูนิกซ์

* 1. ประเภทกราฟิก (Graphic user interface. GUI)

ระบบการติดต่อสื่อสารระหว่างผู้ใช้งานกับเครื่อง[คอมพิวเตอร์](http://www.comgeeks.net/)โดยผ่านทางภาพหรือแผนภูมิที่เข้าใจได้ง่ายแทนการพิมพ์คำสั่งโดยตรง เพื่อให้คอมพิวเตอร์ทำงานตามที่ต้องการ เช่น การใช้เมาส์กดเลือกไอคอน (Icon) หรือปุ่มคำสั่งที่ต้องการ หรือการเลือกคำสั่งตามรายการที่อยู่ในเมนูประเภทต่างๆ ซึ่ง[ซอฟต์แวร์](http://www.comgeeks.net/software/)เตรียมไว้ให้แล้ว

**2.3.2 ประเภทชุดคำสั่งของเชลล์**

1. บอร์นเชลล์ (Bourne shell, sh)

เป็นโปรแกรมเชลล์มาตรฐานการในการติดต่อกับผู้ใช้ผ่านทางบรรทัดคำสั่ง บนระบบปฏิบัติการยูนิกซ์ เวอร์ชัน 7 โปรแกรมนี้แทนที่ทอมสันเชลล์ (Thomson shell) ในยูนิกซ์รุ่นก่อนหน้านั้นซึ่งใช้ชื่อโปรแกรมเอสเอช (sh) เหมือนกัน ปัจจุบันบอร์นเชลล์ยังเป็นเชลล์พื้นฐานที่มีในยูนิกซ์ทุกตัว และยังสามารถย้ายเชลล์สคริปต์ไปยังยูนิกซ์ระบบอื่นได้โดยไม่ต้องแก้ไข ตัวโปรแกรมมักติดตั้งอยู่ที่ /bin/sh ในระบบยูนิกซ์ส่วนใหญ่

* 1. ซีเชลล์ (C shell, csh)

เป็นเชลล์ที่พัฒนาขึ้นมาหลังจากบอร์นเชลล์ที่มีรูปแบบคำสั่งและไวยกรณ์เหมือนกับภาษาซี มีฟังก์ชั่นการทำงานที่สะดวก และยังสามารถควบคุมการไหลของข้อมูลได้ดีกว่าบอร์นเชลล์ ตัวโปรแกรมมักติดตั้งอยู่ที่ /bin/csh

|  |  |
| --- | --- |
| *#!/bin/sh* if [ $days -gt 365 ] then   echo This is over a year. fi | *#!/bin/csh* if ( $days > 365 ) then   echo This is over a year. endif |

**รูปที่ 2.6** เปรียบเทียบการใช้คำสั่งแบบมีเงื่อนไข ระหว่างบอร์นเชลล์ (ซ้าย) และซีเชลล์ (ขวา)

|  |  |
| --- | --- |
| *#!/bin/sh* i=2 j=1 while [ $j -le 10 ] do   echo '2 \*\*' $j = $i   i=`expr $i '\*' 2`   j=`expr $j + 1` done | *#!/bin/csh* set i = 2 set j = 1 while ( $j <= 10 )   echo '2 \*\*' $j = $i   @ i \*= 2   @ j++ end |

**รูปที่ 2.7** เปรียบเทียบการตั้งค่าตัวแปลระหว่างบอร์นเชลล์ (ซ้าย) และซีเชลล์ (ขวา)

* 1. คอร์นเชลล์ (Korn shell, ksh)

เป็นเชลล์ที่พัฒนามาจากต้นแบบของบอร์นเชลล์และซีเชลล์ สามารถทำงานในฟังก์ชันของบอร์นเชลล์ได้ทุกอย่าง ทำให้การเขียนสคริปต์นั้นทำได้ง่ายและรัดกุมยิ่งขึ้น สามารถนำคำสั่งที่ใช้งานไปแล้วกลับมาทำใหม่ได้ ถือว่าคอร์นเชลล์เป็นการรวบรวมเอาข้อดีของบอร์นเชลล์และซีเชลล์มาไว้ด้วยกัน แต่คอร์นเชลล์ไม่ได้มีในยูนิกซ์ทุกตัว

* 1. แบช (Bourne again shell, bash)

เป็นเชลล์ยูนิกซ์ที่ถูกพัฒนามาจากการเอาบอร์นเชลล์นำมาพัฒนาใหม่ โดย

ยูนิกซ์ (Unix) ส่วนใหญ่ใช้แบชเป็นเชลล์มาตรฐาน รูปแบบของคำสั่งของแบชครอบคลุมคำสั่งทั้งหมดของบอร์นเชลล์ ดังนั้นสคริปต์ที่เขียนด้วยบอร์นเชลล์ส่วนใหญ่จึงสามารถรันในแบชได้โดยไม่ต้องมาการแก้ไข จะมียกเว้นก็เช่น สคริปต์ที่เรียกใช้ตัวแปรพิเศษในบอร์นเชลล์ หรือใช้คำสั่งภายในของบอร์นเชลล์ รูปแบบคำสั่งในแบชยังได้รับแนวความคิดจาก[คอร์นเชลล์](https://th.wikipedia.org/w/index.php?title=%E0%B8%84%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%99%E0%B9%80%E0%B8%8A%E0%B8%A5%E0%B8%A5%E0%B9%8C&action=edit&redlink=1)และ[ซีเชลล์](https://th.wikipedia.org/w/index.php?title=%E0%B8%8B%E0%B8%B5%E0%B9%80%E0%B8%8A%E0%B8%A5%E0%B8%A5%E0%B9%8C&action=edit&redlink=1) เวลาใช้เป็นเชลล์ทางบรรทัดคำสั่ง แบชจะเติมชื่อโปรแกรม ชื่อไฟล์ ชื่อตัวแปร ให้ครบโดยอัตโนมัติ เมื่อผู้ใช้กดปุ่มแท็บ(TAB)

**2.3.3 คำสั่งพื้นฐานของเชลล์**

* คำสั่ง ls คือ คำสั่งที่ใช้สำหรับแสดงแฟ้มข้อมูล ย่อมาจากคำว่า “list”

ตัวเลือก (Option) ที่มักใช้กันใน ls คือ

* -l จะแสดงผลลัพธ์แบบ Long Format ซึ่งจะแสดงการอนุญาตเข้าใช้งานของแฟ้มข้อมูลด้วย
* -a จะแสดงแฟ้มข้อมูลทั้งหมด

|  |
| --- |
| **ls** [option] [file] |

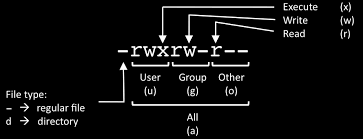
**รูปที่ 2.8** การใช้คำสั่ง ls

* คำสั่ง cd  คือ คำสั่งที่ใช้สับเปลี่ยนสารบบ (Change directory) ของระบบยูนิกซ์และลีนุ๊กซ์
* คำสั่ง cp คือ เป็นคำสั่งที่ใช้สำหรับสำเนาแฟ้มข้อมูล เหมือนกับคัดลอก (Copy)

|  |
| --- |
| **cp** [source] [target] |

**รูปที่ 2.9** การใช้คำสั่ง cp

* คำสั่ง cat คือ ใช้สำหรับดูข้อมูลในแฟ้มข้อมูล และแสดงผลออกทางหน้าจอ (Standard Output) มาจากคำว่า “Concatinate”
* คำสั่ง chmod คือ ใช้เปลี่ยนสิทธิของแฟ้มข้อมูล ได้แก่ เจ้าของ (Owner) คนในกลุ่ม (Group) คนอื่น (Other) โดยแต่ละกลุ่มสามารถแบ่งกำหนดสิทธิได้ 3 ประเภทดังนี้
  + read (r)
  + write (w)
  + execute (x)

[](https://danflood.com/wp-content/uploads/2013/12/permissions1.png)

**รูปที่ 2.10** แสดงกลุ่มของการเข้าถึงแฟ้มข้อมูล

สิทธิการเข้าถึงแฟ้มข้อมูลบนยูนิกซ์สามารถแสดงด้วยตัวอักษร หรือ เลขฐานแปด (Octal digits) ตามตารางที่ 2.1

**ตารางที่ 2.1** แสดงสิทธิการเข้าถึงแฟ้มข้อมูล

|  |  |  |
| --- | --- | --- |
| **Octal** | **Symbol** | **Permission** |
| 0 | --- | No Permissions |
| 1 | --x | Execute |
| 2 | -w- | Write |
| 3 | -wx | Write and Execute |
| 4 | r-- | Read |
| 5 | r-x | Read and Execute |
| 6 | rw- | Read and Write |
| 7 | rwx | Read, Write and Permissions |

วิธีการใช้งานคำสั่ง chmod นั้นสามารถใช้งานได้ 2 แบบคือ การใช้แบบตัวอักษร (Symbols) และ การใช้งานแบบตัวเลขฐานแปด (Octal values)

* การใช้คำสั่งแบบตัวอักษร นั้นมีดังนี้
* u หมายถึง ผู้เป็นเจ้าของแฟ้ม (User)
* g หมายถึง ผู้เป็นเจ้าของแฟ้ม (Gruop)
* o หมายถึง บุคคลอื่นๆ
* a หมายถึง ทุกๆ กลุ่ม
* r หมายถึง สิทธิในการอ่าน
* w หมายถึง สิทธิในการเขียนและแก้ไข
* w หมายถึงสิทธิในการเรียกใช้งานหรือค้นหา (ในกรณีของสารบบ)

|  |
| --- |
| **chmod** [option] [symbols] [file] |

**รูปที่ 2.11** การใช้คำสั่ง chmod แบบตัวอักษร

* การใช้งานแบบตัวเลขฐานแปด เป็นการจัดเรียงตัวเองขออกเป็ฯ 3 กลุ่ม โดยจะให้สิทธิเข้าถึงแฟ้มข้อมูลตารางที่ 2.1

|  |
| --- |
| **chmod** [option] [octal values] [file] |

**รูปที่ 2.12** การใช้คำสั่ง chmod แบบตัวเลขฐานแปด

* คำสั่ง df คือ คำสั่งที่ใช้ในการตรวจสอบการใช้พื้นที่ของฮาร์ดดิกส์ (Hardisk)
* คำสั่ง pwd คือ คำสั่งที่ใช้แสดงสารบบ (Directory) ปัจจุบัน มาจากคำว่า print work directory
* คำสั่ง kill เป็นคำสั่งที่ใช้ยกเลิกโปรเซส (Process) ของระบบ

|  |
| --- |
| **kill** [option] [ID process] |

**รูปที่ 2.13** รูปแบบการใช้งานคำสั่ง kill

* คำสั่ง mkdirคือ คำสั่งที่ใช้สำหรับสร้างสารบบ มาจากคำว่า make directory
* คำสั่ง mv คือ คำสั่งที่ไว้ใช้ย้ายแฟ้มข้อมูลและสารบบ รวมถึงการเปลี่ยนชื่อด้วย มาจากคำว่า move

|  |
| --- |
| **mv** [source] [target] |

**รูปที่ 2.14** รูปแบบการใช้คำสั่ง mv

* คำสั่ง man คือ คำสั่งที่ใช้แสดงและอธิบายการใช้คำสั่งต่างๆ ของระบบยูนิกซ์

|  |
| --- |
| **man** [command] |

**รูปที่ 2.15** รูปแบบการใช้งานคำสั่ง man

* คำสั่ง ssh คือ โปรแกรมสำหรับเข้าสู่ระบบและคำสั่งที่เครื่องปลายทางได้ โดยไม่จำเป็นต้องไปใช้งานที่หน้าจอคอนโซลของเครื่อง จุดประสงค์หลักของโปรแกรมคือทำหน้าที่แทนโปรแกรมประเภท rlogin, rsh หรือ telnet โดยจะมีการเข้ารหัสข้อมูล (encrypted) เพื่อความปลอดภัยของข้อมูลที่ส่งระหว่าง SSH Client และ SSH Server

**2.3.4 การเขียนเชลล์สคริปต์**

เชลล์สคริปต์ คือ การนำคำสั่งของลีนุ๊กส์มาเรียงต่อกันตามให้ทำงานตามที่ต้องการ โดยอาจจะมีการเพิ่มเติมฟังก์ชั่นขึ้นมา หรือโดยสามารถเขียนในโปรแกรมเขียนโปรแกรม (Editor) ได้ โดยในลินุ๊กซ์จะมีโปรแกรมเขียนภาษา ที่เป็นพื้นฐานให้หลานตัว เช่น วีไอ (vi), นาโน (namo), จีอิดิท (gedit) เป็นต้น

การเขียนเชลล์สคริปท์จะต้องเขียนตามชนิดของเชลล์ เพราะเชลล์แต่ละชนิดมีไวยกรณ์ที่แตกต่างกัน หากเขียนเชลล์สคริปต์ไม่ตรงตามเชลล์ อาจจะทำให้ตัวสคริปต์ที่เขียนใช้งานไม่ได้

**2.4 ระบบการควบคุมรุ่นของซอฟต์แวร์และกิต**

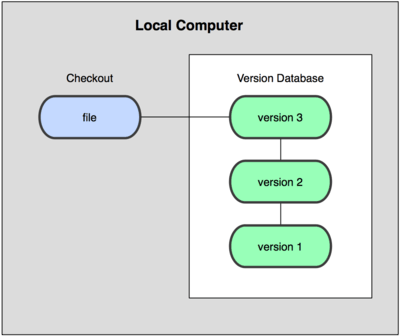
**2.4.1 ความหมายของระบบควบคุมรุ่นของซอฟต์แวร์**

ระบบควบคุมรุ่นของซอฟต์แวร์ (Version control system) คือ เป็นระบบที่จัดเก็บการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์หนี่ง หรือหลายไฟล์ เพื่อที่คุณสามารถเรียกรุ่นในรุ่นหนึ่งกลับมาดูได้เสมอ เราสามารถใช้ ระบบควบคุมรุ่นของซอฟต์แวร์กับไฟล์ชนิดไหนก็ได้ ระบบควบคุมรุ่นของซอฟต์แวร์สามารถเปรียบเทียบการแก้ไขไฟล์ที่เกิดขึ้นในอดีต สามารถดูได้ว่าใครเป็นคนแก้ไขไฟล์ แก้ไขเมื่อไร แก้อะไรไปบ้าง เป็นต้น และยังสามารถกู้ไฟล์ที่ลบไปแล้วได้อย่างง่ายดาย

**2.4.2 ประเภทของระบบการควบคุมรุ่นซอฟต์แวร์**

1. **ระบบการควบคุมรุ่นของซอฟต์แวร์แบบ local**

เป็นการคัดลอกไฟล์ที่ต้องการจะเก็บไว้ในไดเรกทอรี่ใหม่ อาจจะตั้งชื่อไดเรกทอรี่ด้วยวันที่ วิธีนี้เป็นวิธีที่แพร่หลาย เพราะสามารถทำได้ง่ายแต่ในขณะเดียวกันเป็นวิธีที่เกิดขึ้นผิดผลาดได้ง่ายเหมือนกัน

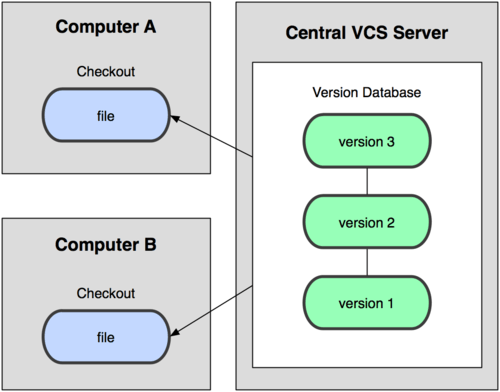


**รูปที่ 2.16** ระบบ Version control แบบ local

* 1. **ระบบควบคุมรุ่นของซอฟต์แวร์แบบรวมศูนย์**

ระบบถูกพัฒนาเนื่องจากนักพัฒนาต้องการพัฒนาร่วมกับนักพัฒนาคนอื่น จึงเกิดระบบนี้ขึ่นมา เรียกว่า  Centralived Version Control System (CVCSs) หรือระบบ Version Control System แบบรวมศูนย์กลาง ซอฟตฺ์แวร์ที่ใช้ระบบนี้ได้แก่ CVS, Subversion และ Perforce ซึ่งมีเซิฟเวอร์กลางที่เก็บไฟล์ทั้งหมดเองไว้ที่เดียวกัน เพื่อให้ผู้ใช้หลายๆ คนสามารถต่อเข้ามาเพื่อดึงข้อมูลหรือไฟล์จากศูนย์กลางออกไปแก้ไขที่เครื่องของตัวเองได้ ระบบนี้ถูกนำมาใช้หลายปี

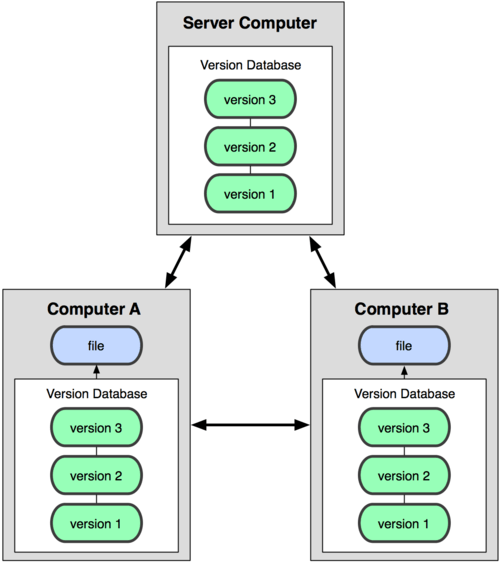
ระบบแบบศูนย์รวมมีข้อได้เปรียบกว่าระบบ local ในหลายๆ ด้าน เช่น ทุกคนสามารถรู๋ได้ว่าคนอื่นในโปรเจคกำลังทำอะไรอยู่ สามารถจัดการข้อมูลได้ที่ศูนย์กลางที่เดียว ทำให้จัดการข้อมูลได้ง่ายกว่า แต่ระบบนี้มีจุดอ่อนตรงที่การรวมศูนย์กลาง ทำให้เป็นจุดเดียวที่เซิฟเวอร์ หมายความว่าถ้าเซิฟเวอร์นั้นเกิดปัญหา ก็จะไม่มีใครที่สามารถทำงานร่วมกันได้ หรือบันทึกการเปลี่ยนแปลงงานที่กำลังทำอยู่ไปที่เซิฟเวอร์ได้เลย หรือถ้าหน่วยความจำสำรองของเครื่องเซิฟเวอร์นั้นเกิดปัญหาและไม่มีการสำรองข้อมูลไว้ หมายความว่าข้อมูลทั้งหมดที่อยู่บนเครื่องเซิฟเวอร์จะหายไปทั้งหมด เหลือเพียงงานที่อยู่บนเครื่องแต่ละเครื่องเท่านั้น



**รูปที่ 2.17** ระบบควบคุมรุ่นของซอฟต์แวร์ แบบรวมศูนย์

1. **ระบบควบคุมรุ่นของซอฟต์แวร์ แบบกระจายศูนย์**

ชื่อเต็มของระบบนี่คือ Distributed version control system (DVCSs) ซอฟต์แวร์ที่ใช้ระบบนี่คือ Git, Mercurial, Bazzar หรือ Darcs) ในระบบงานแบบนี้แต่ละคนไม่ได้คัดลอกงานล่าสุดเท่านั้น แต่ได้คัดลอก repository มาเลย หมายความว่า ถึงแม้ว่าเซิฟเวอร์จะเสียหรือมีปัญหา ไคลแอนด์ก็ยังสามารถทำงานร่วมกันได้ต่อไป และ repository ของเครื่องไคลแอนด์ยังสามารถคัดลอกกลับไปที่เครื่องเซิฟเวอร์เพื่อกู้ข้อมูลได้อีกด้วย

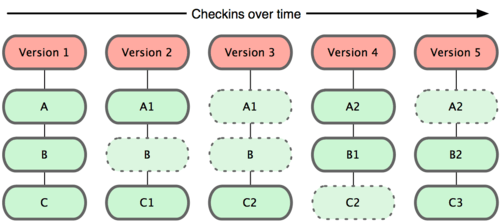


**รูปที่ 2.18** ระบบควบคุมรุ่นของซอฟต์แวร์ แบบกระจายศูนย์

**2.4.3 ความหมายของกิต**

**กิต (GIT)** คือ เป็นซอฟต์แวร์ที่ใช้ในระบบควบคุมรุ่นของซอฟต์แวร์ (Version control system) โดยเริ่มแรกถูกพัฒนามาใช้สำหรับลีนุ๊กซ์ โดยลินุส โตร์วัลดล์ ไดเรกทอรี่ของกิตทุกตัวมีการเก็บประวัติทั้งหมด ปัจจุบันดูแลโดย ฮุนิโอ ฮามาโน กิตเป็นซอฟต์แวร์ฟรีและแจกจ่ายในลักษณะของ GPL (GNU General Public Licence)

การทำงานของกิต จะมองข้อมูลเป็นเหมือนภาพถ่าย (Sanpshot) ของระบบไฟล์ขนาดเล็กๆ ทุกครั้งที่มีการบันทึกสถานภาพของโปรเจค (Commit) ลงในกิต กิตจะทำการถ่ายภาพของไฟล์ทั้งหมดในตอนนั้นและบันทึกการเปลี่ยนแปลง เพื่อให้การจัดเก็บมีประสิทธิภาพ ถ้าไฟล์ไม่ได้มีกานเปลี่ยนแปลง กิตจะไม่บันทึกไฟล์นั้นอีกครั้ง เพียงแต่ทำการเชี่อมโยงไปยังไฟล์เดิมที่เคยถูกบันทึกเอาไว้อยู่แล้ว



**รูปที่ 2.19** การเก็บข้อมูลแบบภาพถ่ายของกิต