บทที่ 5

การทำงานของฟังก์ชันหลักและฟังก์ชันย่อยที่พัฒนาขึ้น

ในการทำงานของฟังก์ชันต่าง ๆ ที่พัฒนาขึ้น จำเป็นต้องมีโครงสร้าง 2 โครงสร้าง คือ

1. funcMap

ประกอบด้วยค่าข้อมูล 2 ฟิลด์ คือ ชื่อโปรเซสที่ผู้ใช้ระบุให้สร้างขึ้นเพื่อประมวลผลแบบจวบกัน และ ตำแหน่งของฟังก์ชันที่จะถูกประมวลผล (ดูรายละเอียดโครงสร้างที่ภาคผนวก)

ผู้ใช้จำเป็นต้องระบุค่าลงในโครงสร้างนี้ก่อนเรียกซีเอสพีพาร์ฟังก์ชัน สมาชิกลำดับสุดท้ายของโครงสร้างนี้ต้องมีค่าข้อมูลในฟิลด์ที่ 1 เป็น NULL เพื่อบอกว่าการกำหนดค่าจบสิ้นลง

2. procMap

ประกอบด้วยค่าข้อมูล 2 ฟิลด์ คือ ชื่อโปรเซสแต่ละโปรเซสที่ถูกสร้างขึ้นพร้อมด้วยค่าพารามิเตอร์ (ถ้ามี) และ หมายเลขโปรเซส (ดูรายละเอียดโครงสร้างที่ภาคผนวก)

โครงสร้างนี้ใช้ในการทำงานของฟังก์ชันหลักทั้ง 3 เท่านั้น ผู้ใช้มิได้ใช้โครงสร้างนี้

ฟังก์ชันย่อย

เพื่อเอื้อให้ฟังก์ชันหลักทั้ง 3 ทำงานได้อย่างมีประสิทธิภาพ จำเป็นต้องมีฟังก์ชันย่อยจำนวนหนึ่งที่ฟังก์ชันหลักทั้ง 3 เรียกใช้

ฟังก์ชันย่อยที่พัฒนาขึ้น มีดังนี้ คือ

1. ฟังก์ชันแฮฟพาเรน (Have_paren function)

ทำหน้าที่ตรวจสอบว่าชื่อโปรเซสมีเครื่องหมายวงเล็บหรือไม่

2. ฟังก์ชันเก็ทพาร์ม (Get_parm function)

ทำหน้าที่ดึงค่าพารามิเตอร์ที่อยู่ในเครื่องหมายวงเล็บของชื่อโปรเซส

3. ฟังก์ชันเสิร์ชเทเบิล (Search_table function)

ทำหน้าที่เปรียบเทียบชื่อโปรเซสกับโครงสร้าง funcMap หรือ procMap แล้วแต่กรณี

4. ฟังก์ชันคอนว์พาร์ม (Conv_parm function)

ทำหน้าที่คำนวณพารามิเตอร์ที่อยู่ในรูปนิพจน์ให้เป็นค่าข้อมูลเพียงค่าเดียว

5. ฟังก์ชันทรานซเฟอร์ (Transfer function)

ทำหน้าที่ส่งผ่านโครงสร้าง procMap จากโปรเซสพ่อ (parent process) ให้โปรเซสลูก (child process) ทั้งหมดเพื่อใช้เปรียบเทียบชื่อโปรเซสที่ต้องการติดต่อด้วย

6. ฟังก์ชันคิลไชล์ด (Killchild function)

ทำหน้าที่หยุดการประมวลผลของโปรเซสลูกทุกโปรเซส

7. ฟังก์ชันเช็คพีเนม (Check_pname function)

ทำหน้าที่ตรวจสอบความถูกต้องของชื่อโปรเซส โดยจะทำงานร่วมกับฟังก์ชันแฮฟพาร์มและฟังก์ชันเก็ทพาร์ม

8. ฟังก์ชันพาร์เซิร์ฟเวอร์ (ParServer function)

ทำหน้าที่ตรวจสอบว่าโปรเซสลูกยังประมวลผลอยู่หรือไม่

9. ฟังก์ชันเซ็นดายน์ (SendDie function)

ทำหน้าที่ส่งข้อมูลให้โปรเซสพ่อรับรู้ว่าโปรเซสลูกโปรเซสใดพร้อมที่จะหยุดการประมวลผล

10. ฟังก์ชันอาร์เอ็มไอพีซี (RmIpc function)

ทำหน้าที่ลบคิวข่าวสารของโปรเซสที่หยุดการประมวลผลแล้วออกจากหน่วยความจำ

11. ฟังก์ชันฟายอีแอลอีโน (FindEleNo function)

ทำหน้าที่ตรวจสอบว่าชื่อโปรเซสที่ต้องการติดต่อด้วยอยู่ในสมาชิกที่เท่าไรของโครงสร้าง procMap

12. ฟังก์ชันแสกนบาวด์ (ScanBound function)

ทำหน้าที่หาขอบเขตล่าง (Lower Boundary) และ ขอบเขตบน (Upper Boundary) ของพารามิเตอร์ที่ระบุในชื่อโปรเซส

ฟังก์ชันแฮฟพาเรน

รูปแบบ

```
int  Have_paren(parm)
char parm[];
```

ฟังก์ชันนี้จะหาเครื่องหมายวงเล็บเปิดในชื่อโปรเซสที่ถูกระบุในพารามิเตอร์ parm และ ส่งค่าตำแหน่งที่เครื่องหมายนั้นอยู่กลับ ถ้าไม่พบ คือ พบเครื่องหมายวงเล็บปิด หรือ หาจนกระทั่งถึงตัวสุดท้ายของชื่อแล้วไม่พบ จะส่งค่า ERROR กลับ

ฟังก์ชันเก็ทพาร์ม

รูปแบบ

```
int  Get_parm(parm, ele_no)
char parm[];
int  ele_no;
```

ฟังก์ชันนี้จะดึงพารามิเตอร์ที่ถูกระบุในเครื่องหมายวงเล็บ ที่อยู่ในชื่อโปรเซสที่ถูกระบุในพารามิเตอร์ชื่อ parm โดยจะเริ่มหาตั้งแต่ตำแหน่งที่ถูกระบุโดยพารามิเตอร์ชื่อ ele_no ในการดึงพารามิเตอร์นี้จะเปรียบเทียบว่าแต่ละตำแหน่งของพารามิเตอร์เป็นตัวเลขหรือไม่

ค่าที่เป็นไปได้ที่ฟังก์ชันนี้จะส่งกลับ คือ

ก) ตัวเลขจำนวนเต็ม ในกรณีที่ค่าพารามิเตอร์เป็นตัวเลขโดด ๆ

ข) NO_PARM ในกรณีที่ไม่มีค่าพารามิเตอร์ระหว่างเครื่องหมายวงเล็บเปิด และเครื่องหมายวงเล็บปิด

ค) NEED_CONV ในกรณีที่ค่าพารามิเตอร์เครื่องหมายที่ใช้ในการคำนวณปนอยู่

ง) ERROR ในกรณีที่ไม่มีเครื่องหมายวงเล็บปิด

ฟังก์ชันเสิร์ชเทเบิล

รูปแบบ

```
int  Search_table(field_no, parm)
int  field_no;
char parm[];
```

ฟังก์ชันเสิร์ชเทเบิลจะเปรียบเทียบค่าที่ระบุในพารามิเตอร์ชื่อ parm กับค่าในโครงสร้าง funcMap ในกรณีที่ค่าในพารามิเตอร์ field_no เป็น FLD_ONE (ซึ่งกำหนดให้มีค่าเท่ากับ 1) และ เปรียบเทียบกับค่าในโครงสร้าง procMap ในกรณีที่ค่าในพารามิเตอร์ field_no เป็น FLD_TWO หรือ FLD_THREE

ถ้าค่าของ field_no เป็น FLD_TWO ฟังก์ชันเสิร์ชเทเบิลจะเปรียบเทียบกับค่าในฟิลด์ชื่อ procName แต่ถ้าเป็น FLD_THREE จะเปรียบเทียบกับค่าในฟิลด์ชื่อ procId

ค่าที่ฟังก์ชันนี้จะส่งกลับ มีได้ดังนี้ คือ

ก) เลขจำนวนเต็ม ในกรณีที่พบค่าที่ต้องการเปรียบเทียบในโครงสร้างดังกล่าว

ข) ERROR ในกรณีที่ไม่พบค่าที่ต้องการเปรียบเทียบ

ฟังก์ชันคอนว์พาร์ม

รูปแบบ

```
int  Conv_parm(parm, ele_no)
char parm[];
int  ele_no;
```

ฟังก์ชันนี้มีการทำงานดังนี้ คือ

1) หาหมายเลขโปรเซสของโปรเซสปัจจุบันแล้วแปลงเป็นชนิดตัวอักษร แล้วเรียกฟังก์ชันเสิร์ชเทเบิลโดยให้ค่า field_no เป็น FLD_THREE และ ส่งค่าหมายเลขโปรเซสที่แปลงชนิดแล้วไปด้วย

2) ดึงชื่อโปรเซสที่อยู่คู่กับหมายเลขโปรเซสในขั้นตอน 1) (โดยอาศัยค่าที่ส่งกลับมาจากฟังก์ชันเสิร์ชเทเบิล) มาหาตำแหน่งของเครื่องหมายวงเล็บเปิด เพื่อหาว่าโปรเซสนั้นมี subscript เป็นเท่าไร

3) ดึงแต่ละตำแหน่งของพารามิเตอร์ที่อยู่ในเครื่องหมายวงเล็บในชื่อที่ถูกระบุโดยพารามิเตอร์ parm มาดูว่าเป็นตัวเลข หรือ ตัวอักษร (เป็นตัวถูกดำเนินการ (Operand)) หรือ เครื่องหมาย บวก ลบ คูณ หาร (เป็นตัวดำเนินการ (Operator)) ทั้งนี้เพื่อคำนวณนิพจน์ให้อยู่ในรูปของเลขจำนวนเดียว (ในกรณีที่ตัวถูกดำเนินการเป็นตัวอักษร จะถูกแทนที่ด้วยค่า subscript ที่ได้จากขั้นตอน 2)) และ ส่งค่านั้นกลับหลังจากที่คำนวณทั้งนิพจน์เสร็จ

4) ในกรณีที่นิพจน์ในเครื่องหมายวงเล็บไม่ถูกวากยสัมพันธ์ คือ ไม่อยู่ในรูปแบบ "ตัวถูกดำเนินการ ตัวดำเนินการ ตัวถูกดำเนินการ" จะหยุดการคำนวณ และ ส่งค่า ERROR กลับ

ฟังก์ชันทรานซเฟอร์

รูปแบบ

```
int  Transfer(from, pid)
int  from, pid;
```

ฟังก์ชันทรานซเฟอร์มีขั้นตอนการทำงาน คือ

1) จะเปรียบเทียบว่าค่าในพารามิเตอร์ชื่อ from มีค่าเป็น CHILD หรือไม่ ถ้าใช่ จะทำขั้นตอนที่ 2) มิฉะนั้น จะทำขั้นตอนที่ 3)

2) เรียก msgget system call เพื่อหาหมายเลขประจำคิวข่าวสารที่มีค่าคีย์เหมือนค่าพารามิเตอร์ชื่อ pid จากนั้นจะเรียก msgrcv system call เพื่อรอรับค่าข้อมูลจากโปรเซสพ่อ (ชนิดของข่าวสารมีค่าเท่ากับหมายเลขโปรเซสของโปรเซสพ่อ) เมื่อได้รับค่าข้อมูลแล้วจะทำการคัดลอกค่าข้อมูลนั้นเก็บไว้ที่โครงสร้าง procMap ของตน และ ส่งข้อมูล ("ACK") กลับให้โปรเซสพ่อรับทราบ

3) คัดลอกค่าข้อมูลจากโครงสร้าง procMap มาเก็บไว้ที่ตัวแปรตัวหนึ่ง แล้วดึงค่า procId ในสมาชิกที่ 1 จากโครงสร้าง procMap และ เรียก msgget system call โดยให้คีย์มีค่าเท่ากับค่า procId จากนั้นจะเรียก msgsndsystem call เพื่อส่งค่าข้อมูลที่

คัดลอกไว้แล้วนั้น และ กำหนดชนิดของข่าวสารให้มีค่าเท่ากับหมายเลขโปรเซสของตน (ของพ่อ) และ รอรับค่า("ACK") ตอบกลับจากโปรเซสลูก จากนั้น จะกระทำในลักษณะเดียวกันกับprocId ในสมาชิกที่ 2, 3, ... ไปจนกระทั่งครบโครงสร้าง procMap

ค่าที่เป็นไปได้ที่ฟังก์ชันนี้จะส่งกลับ คือ

ก) SUCCESS ในกรณีที่การถ่ายทอดข้อมูลเป็นไปอย่างเสร็จสมบูรณ์

ข) ERROR ในกรณีที่มีข้อผิดพลาดบางประการเกิดขึ้น อันเนื่องมาจาก msgget, msgrcv หรือ msgsnd system call

ฟังก์ชันคิลไชล์ด

รูปแบบ

```
void Killchild();
```

ฟังก์ชันนี้จะดึง procId ในแต่ละสมาชิกของโครงสร้าง procMap มาใช้ในการเรียก kill system call เพื่อส่ง interrupt signal ให้โปรเซสที่มีหมายเลขโปรเซสนั้น ๆ หยุดการประมวลผล จากนั้น จะเรียกฟังก์ชันอาร์เอ็มไอพีซีโดยส่งหมายเลขโปรเซสของตนเป็นพารามิเตอร์เพื่อประมวลผลต่อไป

ฟังก์ชันเช็คพีเนม

รูปแบบ

```
int  Check_pname(pname, parm_flag, int_parm)
char *pname[];
int  *parm_flag;
int  *int_parm;
```

ฟังก์ชันเช็คพีเนมมีขั้นตอนการทำงาน ดังนี้

1) กำหนดค่าเริ่มต้นให้ตัวแปรขอบเขตล่าง และ ตัวแปรขอบเขตบนเป็น -1

2) เรียกฟังก์ชันแฮฟพาเรนเพื่อหาตำแหน่งของเครื่องหมายวงเล็บเปิด ถ้าค่าที่ได้กลับมาน้อยกว่าหรือเท่ากับ 0 จะส่งค่า ERROR กลับ มิฉะนั้นจะเรียกฟังก์ชันเก็ทพาร์ม ถ้าค่าที่ได้กลับมาเป็น NO_PARM จะกำหนดให้ค่าของพารามิเตอร์ parm_flag เป็น FALSE และ ส่งค่า SUCCESS กลับ แต่ถ้าเป็น ERROR จะส่งค่า ERROR กลับ ในกรณีที่เป็น NEED_CONV จะทำขั้นตอน 3)

3) เรียกฟังก์ชันแสกนบาวด์ ถ้าค่าที่ได้กลับมาเป็น ERROR จะส่งค่า ERROR กลับ มิฉะนั้น จะกำหนดให้ค่าของพารามิเตอร์ parm_flag เป็น TRUE และ ส่งค่า SUCCESS กลับ

<u>ฟังก์ชันพาร์เซิร์ฟเวอร์</u>

<u>รูปแบบ</u>

```
int  ParServer();
```

ฟังก์ชันนี้จะดึงค่า procId แต่ละสมาชิกของโครงสร้าง procMap มาตรวจสอบว่ามีค่าเป็น 0 หรือไม่ (จะมีค่าเป็น 0 เมื่อฟังก์ชันพาร์เซิร์ฟเวอร์เคยพบว่าโปรเซสนั้นหยุดการประมวลผลไปแล้ว) ถ้าไม่เป็น จะตรวจสอบดูว่าโปรเซสนั้นยังประมวลผลอยู่หรือไม่ ถ้าพบว่ามิได้ประมวลผลอยู่ จะทำการลบคิวข่าวสารที่มีคีย์เท่ากับ procId นั้น ๆ ออกจากหน่วยความจำ และ กำหนดให้ค่า procId นั้น ๆ เป็น 0 และ จะทำเช่นนี้ไปจนครบโครงสร้าง procMap

ในขณะที่ฟังก์ชันนี้ทำงานข้างต้นในแต่ละรอบ จะตรวจสอบว่ามีโปรเซสลูกโปรเซสใดส่งข้อมูลมาก่อนที่จะหยุดการประมวลผลหรือไม่ ถ้ามีจะตอบ ("ACK") กลับไป และ เรียก wait system call เพื่อมิให้โปรเซสลูกที่หยุดการประมวลผลไปนั้นเป็น daemon โปรเซส

ค่าที่ฟังก์ชันนี้ส่งกลับ คือ จำนวนโปรเซสที่ถูกพบว่าหยุดการประมวลผลลงในการทำงานครั้งนี้

ฟังก์ชันเซ็นดายน์

รูปแบบ

```
void SendDie(pid)
int  pid;
```

โปรเซสลูกโปรเซสใดที่กำลังจะหยุดการประมวลผลลง จะต้องเรียกฟังก์ชันนี้เพื่อส่งข้อมูล ("READY") ให้โปรเซสพ่อรับทราบ โดยค่าคีย์ที่ใช้คือหมายเลขโปรเซสของโปรเซสพ่อ เมื่อส่งข้อมูลไปแล้ว ฟังก์ชันนี้จะรอรับข้อมูลตอบกลับ ("ACK") จากโปรเซสพ่อ

ฟังก์ชันอาร์เอ็มไอพีซี

รูปแบบ

```
void RmIpc(pid)
int  pid;
```

ฟังก์ชันนี้จะใช้ค่าพารามิเตอร์ pid ที่ส่งเข้ามาเป็นคีย์ที่ใช้ในการหาหมายเลขคิวข่าวสาร เมื่อได้แล้ว จะเรียก msgctl system call เพื่อลบคิวข่าวสารนั้นออกจากหน่วยความจำ

ฟังก์ชันฟายด์อีแอลอีโน

รูปแบบ

```
int  FindEleNo(pname)
char pname[];
```

ฟังก์ชันนี้มีขั้นตอนการทำงานดังนี้

1) เรียกฟังก์ชันแฮฟพาเรน ถ้าค่าที่ได้กลับมาน้อยกว่าหรือเท่ากับ 0 จะส่งค่า COMM_ERR กลับ มิฉะนั้น จะประมวลผลขั้นตอน 2)

2) เรียกฟังก์ชันเก็ทพาร์ม ถ้าค่าที่ได้กลับมาเป็น ERROR จะส่งค่า COMM_ERR กลับ แต่ถ้าค่าที่ได้เป็น เลขจำนวนเต็ม หรือ NO_PARM จะเรียกฟังก์ชันเสิร์ชเทเบิล โดยค่าของ field_no เป็น FLD_TWO และค่าที่เปรียบเทียบคือค่าพารามิเตอร์ pname และ ค่าที่ได้กลับมาจากฟังก์ชันเสิร์ชเทเบิลจะเป็นค่าที่ส่งกลับไปให้ฟังก์ชันที่เรียกฟังก์ชันนี้ แต่ถ้าหากว่าค่าที่ได้กลับมาจากฟังก์ชันเก็ทพาร์มเป็น NEED_CONV จะทำขั้นตอน 3)

3) เรียกฟังก์ชันคอนว์พาร์มเพื่อคำนวณพารามิเตอร์ในเครื่องหมายวงเล็บในชื่อโปรเซสเป็นเลขจำนวนเดียว แล้วใช้ชื่อโปรเซสที่ส่งเข้ามายังฟังก์ชันนี้แต่พารามิเตอร์เป็นค่าที่คำนวณได้ เป็นค่าที่ส่งไปเปรียบเทียบในฟังก์ชันเสิร์ชเทเบิล ค่าที่ได้กลับมาจะถูกส่งกลับไป

<u>ฟังก์ชันแสกนบาวด์</u>

<u>รูปแบบ</u>

```
int  ScanBound(parm, ele_no)
char parm[];
int  ele_no;
```

ฟังก์ชันนี้จะตรวจสอบว่าพารามิเตอร์ในเครื่องหมายวงเล็บของชื่อโปรเซสที่ถูกระบุโดย parm ถูกต้องตามรูปแบบ "ตัวแปร:ค่าเริ่มต้น..ค่าสุดท้าย" หรือไม่ ถ้าไม่ถูกต้องจะส่งค่า ERROR กลับ แต่ถ้าถูกต้องจะแปลงค่าเริ่มต้น และ ค่าสุดท้ายให้เป็นชนิดตัวเลข และ เก็บไว้ในตัวแปรขอบเขตล่าง และ ตัวแปรขอบเขตบน ตามลำดับ และ ส่งค่า SUCCESS กลับ อย่างไรก็ตาม ค่าเริ่มต้นต้องมีค่าน้อยกว่าหรือเท่ากับค่าสุดท้าย มิฉะนั้นจะส่งค่า ERROR กลับเช่นกัน

ฟังก์ชันหลัก

1. ฟังก์ชันซีเอสพีพาร์

รูปแบบ

```
int  cspPar(param)
char *param[];
```

ฟังก์ชันซีเอสพีพาร์มีขั้นตอนการทำงาน ดังนี้ คือ

1) กำหนดให้จำนวนโปรเซสเป็น 0

2) ดึงชื่อโปรเซสที่ระบุใน param ออกมา 1 ชื่อ และ เรียกฟังก์ชันเช็คพีเนม ถ้าค่าที่ได้กลับมาเป็น ERROR จะเรียกฟังก์ชันคิลไชล์ดและส่งค่า ERROR กลับ มิฉะนั้น จะดึงชื่อโปรเซส (ไม่มีเครื่องหมายวงเล็บและค่าในเครื่องหมายวงเล็บ) เพื่อส่งให้ฟังก์ชันเสิร์ชเทเบิลเปรียบเทียบ โดยให้ค่าพารามิเตอร์ field_no เป็น FLD_ONE ถ้าค่าที่ได้กลับมาน้อยกว่า 0 จะเรียกฟังก์ชันคิลไชล์ดและส่งค่า ERROR กลับ

3) ทำการสร้างโปรเซสลูกขึ้น 1 โปรเซส โดยใช้ fork system call และ ให้โปรเซสลูกเรียกฟังก์ชันทรานซเฟอร์ ถ้าการถ่ายทอดโครงสร้าง procMap เป็นไปอย่างสมบูรณ์ จะประมวลผลฟังก์ชันที่ระบุในฟิลด์ ptr ของโครงสร้าง funcMap ต่อไป และเมื่อกลับมาจากการประมวลผลฟังก์ชันนั้นแล้ว ให้เรียกฟังก์ชันเซ็นดายน์ แล้วจึงหยุดการประมวลผล สำหรับโปรเซสพ่อ จะทำการบันทึกลงในโครงสร้าง procMap ในสมาชิกอันดับถัดจากอันดับที่แล้ว (สำหรับครั้งแรกจะเป็นอันดับ 0) ว่าโปรเซสลูกนั้นชื่ออะไร มีหมายเลขประจำโปรเซสเท่าไร

4) โปรเซสพ่อจะทำขั้นตอน 2) และ 3) ไปจนกระทั่งครบตามที่ระบุใน param จากนั้น จะเรียกฟังก์ชันทรานซเฟอร์เพื่อส่งค่าข้อมูลตามโครงสร้าง procMap ให้โปรเซสลูก จากนั้น จะเรียกฟังก์ชันพาร์เซิร์ฟเวอร์จนกว่าโปรเซสลูกทุกโปรเซสจะหยุดการประมวลผล แล้วจะเรียกฟังก์ชันอาร์เอ็มไอพีซีโดยส่งค่าหมายเลขโปรเซสของตนไป แล้วส่งค่า SUCCESS กลับ (อย่างไรก็ตาม ในการส่งค่าข้อมูลตามโครงสร้าง procMap ให้โปรเซสลูก ถ้าค่าที่ได้เป็น ERROR โปรเซสพ่อจะเรียกฟังก์ชันคิลไชล์ดและส่งค่า ERROR กลับ)

2. ฟังก์ชันซีเอสพีอิน

รูปแบบ

```
int  cspin(s_name, int_value, flag)
char s_name[];
int  *int_value, flag;
```

ฟังก์ชันซีเอสพีอินมีขั้นตอนการทำงาน ดังนี้

1) เรียกฟังก์ชันฟายต์อีแอลอีโนโดยส่ง s_name ไป ถ้าค่าที่ได้กลับมาเป็น ERROR หรือ COMM_ERR จะส่งค่า COMM_ERR กลับ มิฉะนั้น จะดึงค่า procId ในสมาชิกที่หาได้ มาเป็นหมายเลขโปรเซสของโปรเซสต้นทาง

2) หาหมายเลขคิวข่าวสารโดยใช้หมายเลขโปรเซสของตนเป็นคีย์ โดยระบุว่า ถ้าไม่มีคิวข่าวสารนี้อยู่ ให้สร้างขึ้นใหม่ จากนั้น จะตรวจสอบค่าของพารามิเตอร์ flag ที่ส่งเข้ามา ถ้ามีค่าเป็น 0 จะทำขั้นตอน 3) ถ้ามีค่าเป็น IPC_NOWAIT จะทำขั้นตอน 4) มิฉะนั้น จะส่งค่า COMM_ERR กลับ

3) เรียก msgrcv system call โดยใช้หมายเลขโปรเซสของโปรเซสต้นทางเป็นชนิดของข่าวสาร และ ส่ง 0 ไปเป็น flag ถ้าไม่มีข้อมูลส่งมาจะตรวจสอบว่าโปรเซสต้นทางนั้นหยุดประมวลผลไปหรือยัง ถ้ายังจะทำขั้นตอนนี้ไปเรื่อย ๆ จนกว่าจะมีข้อมูลส่งมา แต่ถ้าหยุดประมวลผลแล้ว จะส่งค่า COMM_DIE กลับ เมื่อมีข้อมูลส่งมา จะทำขั้นตอน 5)

4) เรียก msgrcv system call โดยใช้หมายเลขโปรเซสของโปรเซสต้นทางเป็นชนิดของข่าวสาร และ ส่ง IPC_NOWAIT ไปเป็น flag ถ้าไม่มีข้อมูลส่งมาจะตรวจสอบว่าโปรเซสต้นทางนั้นหยุดประมวลผลไปหรือยัง ถ้ายังจะส่งค่า COMM_NOK กลับ แต่ถ้าหยุดประมวลผลแล้ว จะส่งค่า COMM_DIE กลับ สำหรับกรณีที่มีข้อมูลส่งมา จะทำขั้นตอน 5)

5) ทำการคัดลอกข้อมูลที่ส่งมาไว้ในตำแหน่งที่ int_value ชี้อยู่ แล้วเรียก msgget system call โดยใช้หมายเลขโปรเซสของโปรเซสต้นทางเป็นคีย์ จากนั้น เรียก msgsnd system call โดยใช้หมายเลขโปรเซสของตนเป็นชนิดของข่าวสาร เพื่อส่ง ("ACK") ให้โปรเซสต้นทางรับทราบ ถ้าการส่งนี้สมบูรณ์ จะส่งค่า COMM_OK กลับ มิฉะนั้น จะส่งค่า COMM_NOK กลับ

3. ฟังก์ชันซีเอสพีเอาท์

รูปแบบ

```
int  cspOut(d_name, int_value)
char d_name[];
int  *int_value;
```

ฟังก์ชันซีเอสพีเอาท์มีขั้นตอนการทำงาน ดังนี้

1) เรียกฟังก์ชันฟายต์อีแอลอีโนโดยส่ง d_name ไป ถ้าค่าที่ได้กลับมาเป็น ERROR หรือ COMM_ERR จะส่งค่า COMM_ERR กลับ มิฉะนั้น จะดึงค่า procId ในสมาชิกที่หาได้ มาเป็นหมายเลขโปรเซสของโปรเซสปลายทาง

2) ตรวจสอบว่าโปรเซสปลายทางประมวลผลอยู่หรือไม่ ถ้ามิได้ประมวลผลอยู่ จะส่งค่า COMM_DIE กลับ

3) หาหมายเลขคิวข่าวสาร โดยใช้หมายเลขโปรเซสของโปรเซสปลายทางเป็นคีย์ โดยระบุว่าถ้าไม่มีคิวข่าวสารนี้อยู่ให้สร้างขึ้นใหม่ ถ้าหาหมายเลขคิวข่าวสารไม่ได้ จะส่งค่า COMM_NOK กลับ

4) เรียก msgsnd system call โดยใช้หมายเลขโปรเซสของตนเป็นชนิดของข่าวสาร เพื่อส่งค่าข้อมูลในตำแหน่งที่ int_value ชี้อยู่ไปให้โปรเซสปลายทาง ถ้าการส่งข้อมูลเกิดผิดพลาด จะส่ง COMM_NOK กลับ

5) หาหมายเลขคิวข่าวสารที่มีหมายเลขโปรเซสของตนเป็นคีย์ แล้วเรียก msgrcv system call โดยใช้หมายเลขโปรเซสของโปรเซสต้นทางเป็นชนิดของข่าวสาร เพื่อรอรับค่า ("ACK") ตอบกลับมาจากโปรเซสปลายทาง ถ้าการรับนี้สมบูรณ์จะส่ง COMM_OK กลับ มิฉะนั้น จะส่ง COMM_NOK กลับ

เอกสารประกอบการเรียกใช้ฟังก์ชันหลัก

นอกจากไลบรารีฟังก์ชันที่ถูกพัฒนาขึ้นแล้ว ยังมีการจัดทำเอกสารประกอบการเรียกใช้ โดยมีลักษณะเป็นแบบอยู่ในเครื่องคอมพิวเตอร์ (On-line manual) โดยแบ่งเป็น 4 ฉบับ คือ

1. เอกสารประกอบแนวความคิดซีเอสพี
2. เอกสารประกอบการเรียกใช้ฟังก์ชันซีเอสพีพาร์
3. เอกสารประกอบการเรียกใช้ฟังก์ชันซีเอสพีอิน
4. เอกสารประกอบการเรียกใช้ฟังก์ชันซีเอสพีเอาท์