สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การศึกษาใช้งานระบบเรียลไทม์ผ่านการติดต่อแบบอนุกรม

CASE STUDY: USING SERIAL COMMUNICATION ON REAL TIME SYSTEM

บัณฑิต อัมพุชินทร์
มโน กิตติพัฒนาพงศ์

เลขหมู่.....
เลขทะเบียน 59394
วัน,เดือน,ปี 27 ... 2549

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

CASE STUDY:USING SERIAL COMMUNICATION ON REAL TIME SYSTEM

BUNDIT AMPUTCHIN

MANO KITTIPATTANAPONG

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2005

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปัญหาพิเศษ	การศึกษาการใช้งานระบบเรียลไทม์ผ่านการติดต่อแบบอนุกรม CASE STUDY : USING SERIAL COMMUNICATION ON REAL TIME SYSTEM
ชื่อนักศึกษา	นายบัณฑิต อัมพุชินทร์ 45050490 นายมโน กิตติพัฒนาพงศ์ 45050506
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สาขาวิชา	วิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อ.สันธนะ อู่อุดมยิ่ง

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2548

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ.ดร.นันทิกา เบญจเทพานันท์	
กรรมการ	ผศ.สิริลักษณ์ อนันต์สถิตย์สิน	
กรรมการและอาจารย์ที่ปรึกษา	อ.สันธนะ อู่อุดมยิ่ง	

(รองศาสตราจารย์ ดร.วีระ บุญจริง)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปัญหาพิเศษ	การศึกษาการใช้งานระบบเรียลไทม์ผ่านการติดต่อแบบอนุกรม
ชื่อนักศึกษา	นายบัณฑิต อัมพุชินทร์ 45050490
	นายมโน กิตติพัฒนาพงศ์ 45050506
ปริญญา	วิทยาศาสตรบัณฑิต
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สาขาวิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2548
อาจารย์ที่ปรึกษา	อ.สันธนะ อู่อุดมยิ่ง

บทคัดย่อ

โครงงานนี้จัดทำขึ้นเพื่อศึกษาถึงความรู้ทั่วไปเกี่ยวกับระบบปฏิบัติการแบบเรียลไทม์และผลของการใช้งาน ซึ่งได้ติดตั้งระบบปฏิบัติการอาร์ทีลินุกซ์ (อธิบายขั้นตอนโดยละเอียด) แล้วเขียนโปรแกรมเพื่อใช้งานซีพียูแบบปกติ และโปรแกรมแบบเรียลไทม์ รวมทั้งการติดต่อระหว่างโปรแกรมของโปรแกรมทั้งสองแบบ ในการนี้โปรแกรมแบบเรียลไทม์ใช้วิธีการติดต่อสื่อสารแบบอนุกรมเนื่องจากโปรแกรมทางด้านนี้มักมีการติดต่อสื่อสารกับอุปกรณ์ภายนอก โดยโครงงานนี้เน้นศึกษาพฤติกรรมของโปรแกรมผ่านทางเวลาที่ใช้บนระบบที่มีโปรแกรมทั้งสองแบบอีกทั้งมีรูปแบบในการนำหลักการทางสถิติมาช่วยในการวิเคราะห์ผลการทดลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Special Project Title	CASE STUDY : *USING SERIAL COMMUNICATION ON REAL TIME SYSTEM*
Students	Mr.Bundit Amputchin 45050490
	Mr.Mano Kittipattanapong 45050506
Degree	Bachelor of Science
Department	Mathematics and Computer Science, Faculty of Science
Programme	Computer Science
Academic Year	2005
Special Project Adviser	Suntana Oudomying

ABSTRACT

We examined and installed real-time system adopting RT-Linux as well as implementing a serial port communication program and modeling process communication via IPC. We observed performance analysis statistically of the system behavior through experiment of running the real-time process over a system with certain load.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องการศึกษาการใช้งานระบบเรียลไทม์ผ่านการติดต่อแบบอนุกรม สามารถสำเร็จลุล่วงได้ด้วยดีด้วยความช่วยเหลือและความร่วมมือจากหลายๆท่าน คณะผู้จัดทำต้องขอขอบพระคุณบุคคลที่มีส่วนช่วยให้การทำงานครั้งนี้เสร็จไปได้ด้วยดี คือ อ.สันธนะ อู่อุดมยิ่ง อาจารย์ที่ปรึกษาปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำแนวคิดต่างๆในการทำปัญหาพิเศษนี้ ดูแลเอาใจใส่ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของโครงงานปัญหาพิเศษฉบับนี้ และพี่กิตติพงศ์ สังฆมรักษ์ จากศูนย์เทคโนโลยีอิเล็กทรอนิคส์และเทคโนโลยีแห่งชาติที่คอยช่วยตอบปัญหาเกี่ยวกับการติดตั้งระบบปฏิบัติการอาร์ทีลินุกซ์ และการเขียนโปรแกรมแบบเรียลไทม์

นอกจากนี้คณะผู้จัดทำต้องขอขอบบุคคลที่เกี่ยวข้องทุกฝ่าย ที่ทำให้การทำปัญหาพิเศษนี้สำเร็จลุล่วงไปด้วยดี รวมทั้งเพื่อนๆ และพี่ๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2549

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป (ต่อ)

รูปที่ หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

"ระบบเรียลไทม์" ในความหมายทางทั่วไป(การตลาด)หมายถึงระบบที่ทำงานแบบตอบสนองทันที แต่ความจริงแล้วนักพัฒนาโปรแกรมถือว่า คือระบบที่จะทำงานโดยมีข้อกำหนดว่างานที่ทำจะต้องเสร็จก่อนเวลาที่กำหนด[1] ในทางกลับกันถ้าระบบประเมินว่างานนั้น ๆ ไม่สามารถทำเสร็จในเวลาที่กำหนดก็จะไม่ทำ ซึ่งเงื่อนไขดังกล่าวเป็นคุณสมบัติที่สำคัญ[2](มากกว่าการได้ผลลัพธ์ที่ถูกต้อง) หมายความว่า เราอาจต้องการให้คอมพิวเตอร์ทำงานได้ภายในระยะเวลาที่เรากำหนด หรือทำงานที่เวลาที่เรากำหนดโดยความผิดพลาดทางเวลาที่ยอมรับได้อาจจะน้อยมาก อาจจะเป็น 1/1000 วินาที(Millisecond)หรือ น้อยกว่านั้น เพื่อที่จะได้มีการติดต่อกับอุปกรณ์อิเล็คทรอนิคส์ได้อย่างต่อเนื่อง โดยระบบเรียลไทม์ จะต้องสามารถคำนวณได้ว่าจะรับงานนั้นๆเข้าไปทำหรือไม่ด้วยเงื่อนไขว่าจะทำเสร็จก่อนเดดไลน์ (deadline)

ปัจจุบันระบบปฏิบัติการแบบเรียลไทม์ มีขนาดเล็ก และมีประสิทธิภาพสูง สามารถนำไปทดแทนการควบคุมโรงงานที่ใช้เทคนิคการควบคุมด้วยการฮาร์ดโค้ดคำสั่งลงไปในไมโครคอนโทรเลอร์ที่จะรับประกันว่าทุกส่วนของโรงงานจะทำงานสอดคล้องกันภายในเดดไลน์ต่างๆ ซึ่งแม้ว่าจะทำงานได้เร็วแต่ก็เป็นอุปสรรคสำคัญในการบำรุงรักษา และแก้ไข คำสั่งต่างๆ อันเป็นผลให้มีการใช้เครื่องไมโครคอมพิวเตอร์ที่มีระบบเรียลไทม์ในระบบอุตสาหกรรมเพื่อควบคุมเครื่องจักรแทน เนื่องจากสามารถพัฒนาตัวซอฟท์แวร์ตามเทคโนโลยีใหม่ๆได้เร็วโดยการใช้ API ที่ระบบปฏิบัติการมีมาให้

นอกจากนี้ระบบปฏิบัติการที่มีขนาดเล็กกำลังได้รับความสนใจอย่างมาก ระบบปฏิบัติการทั่วไปมีขนาดใหญ่ขึ้น เพื่อเพิ่มความสามารถในการอำนวยความสะดวกต่อผู้ใช้และเพิ่มความสามารถมากขึ้น ซึ่งสอดคล้องกับเทคโนโลยีทางฮาร์ดแวร์ที่มีความศักยภาพมากขึ้น แต่ในขณะเดียวกันขีดความสามารถและทรัพยากรทางฮาร์ดแวร์ก็ได้พัฒนาไปสู่เครื่องคอมพิวเตอร์ขนาดพกพา ซึ่งต้องการระบบปฏิบัติการที่มีขนาดเล็ก โดยในปัจจุบันเครื่องคอมพิวเตอร์ขนาดพกพานั้นได้รับความนิยมกันอย่างแพร่หลาย โดยเราเรียกรูปแบบการใช้ระบบปฏิบัติการขนาดเล็กนี้ว่าระบบแบบฝังตัว (embedded system)

ที่กล่าวมาก็เพื่อชี้ให้เห็นถึงความสัมพันธ์และความน่าสนใจของระบบแบบฝังตัวและระบบแบบเรียลไทม์ โดยระบบที่ใช้ในอุตสาหกรรมมักจะมีลักษณะของระบบฝังตัว ส่วนจะมีคุณสมบัติ

---

[1] เรียลไทม์ไม่ได้หมายความว่ามันต้องตอบสนองทันที แต่หมายถึงต้องตรงเวลา งานที่เป็นเรียลไทม์จะต้องทำให้เสร็จทันเดดไลน์ [7]

[2] ระบบเรียลไทม์เป็นระบบที่ความถูกต้องไม่ได้ขึ้นอยู่กับผลลัพธ์ตามอัลกอริทึมแต่ขึ้นอยู่กับเวลาที่ได้ให้ผลลัพธ์นั้นออกมา [5]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบบเรียลไทม์หรือไม่ เป็นอีกเรื่องหนึ่ง (ในทางกลับกันระบบเรียลไทม์ไม่จำเป็นต้องมีขนาดเล็กเสมอไป)

ถ้าเอ่ยถึงคำว่า "ลินุกซ์" คนที่ใช้คอมพิวเตอร์แทบทุกคน คงรู้จักและคุ้นเคยกับคำนี้ดี เนื่องจากความขึ้นชื่อเรื่องเสถียรภาพในการทำงาน ที่ผ่านๆมาได้มีการนำลินุกซ์ไปใช้ในงานด้าน Server ต่างๆ ทั้งที่จริงๆแล้วไม่ใช่เฉพาะอย่างนี้เท่านั้นที่ลินุกซ์สามารถทำได้ดี งานที่เกี่ยวกับการเชื่อมต่อและควบคุมอุปกรณ์อิเล็คทรอนิคส์ ลินุกซ์ก็สามารถนำมาพัฒนาใช้ได้ดีไม่แพ้ระบบปฏิบัติการอื่นเช่นกัน และโดยเฉพาะอย่างยิ่งที่ลินุกซ์เป็นระบบปฏิบัติการที่สามารถนำมาใช้ได้โดยไม่จำเป็นต้องเสียค่าใช้จ่ายใดๆ จึงมีกลุ่มนักพัฒนาโปรแกรมพัฒนาลินุกซ์ให้สามารถทำงานแบบเรียลไทม์ได้

ในการที่จะสามารถรู้ได้ว่างานนั้นๆจะสามารถทำเสร็จตามเดดไลน์ที่กำหนดหรือไม่นั้นระบบจะต้องควบคุมให้สามารถคำนวณได้อย่างแม่นยำว่างานนั้นๆจะเสร็จเมื่อไหร่ ทุกขั้นตอนเป็นไปอย่างสามารถคำนวณได้ (Deterministic) ซึ่ง 1 ในเทคนิคที่นำมาใช้ได้แก่ การตัดส่วนที่เป็นหน่วยความจำสำรองออกเนื่องจากเป็นส่วนที่ได้รับการออกแบบมาให้ไม่สามารถกำหนดได้ว่าจะได้รับการเลือกให้กลับเข้ามาใช้ CPU ได้อีกเมื่อไร ทำให้ระบบปฏิบัติการแบบเรียลไทม์มีลักษณะเล็กและไม่ซับซ้อน ซึ่งลินุกซ์มีเคอร์เนลแบบสนับสนุนการทำงานแบบเรียลไทม์ เรียกว่า อาร์ทีลินุกซ์(RTLinux) ทางคณะผู้จัดทำจึงเลือกที่จะศึกษาการทำงานของอาร์ทีลินุกซ์

นอกเหนือจากเหตุผลที่กล่าวมาแล้ว ในการที่จะให้ระบบปฏิบัติการอาร์ที ลินุกซ์ ติดต่อกับ(เพื่อควบคุม)อุปกรณ์ภายนอก ทางคณะผู้จัดทำได้เลือกระบบสื่อสารแบบอนุกรม (serial communication) เนื่องจากเป็นพอร์ทที่แพร่หลาย, มีมาตราฐานรองรับ และไม่จำเป็นต้องติดตั้งอุปกรณ์สื่อสารอื่นๆ (เช่น อีเทอร์เนต (Ethernet)) เพิ่มเติม

1.2 วัตถุประสงค์ของการทำปัญหาพิเศษ

1) เพื่อศึกษาความหมายและประโยชน์ของระบบปฏิบัติการแบบเรียลไทม์
2) เพื่อศึกษาการรับ-ส่งข้อมูลแบบเรียลไทม์
3) เพื่อศึกษาการใช้งานระบบปฏิบัติการลินุกซ์ แบบเรียลไทม์

1.3 ขอบเขตของปัญหาพิเศษ

ปัญหาพิเศษนี้เป็นการศึกษาระบบปฏิบัติการแบบเรียลไทม์ แล้วทำการสร้างโมดูลที่เป็นเรียลไทม์สำหรับการรับ-ส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ผ่านพอร์ทอนุกรมและทำการเปรียบเทียบเวลาในการทำงานระหว่างงานแบบปกติกับที่มีงานเรียลไทม์ทำงานอยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้จากการศึกษาระบบปฏิบัติการแบบเรียลไทม์ ว่ามีหลักการในการทำงานอย่างไร

2) ได้รับความรู้จากการศึกษาระบบปฏิบัติการลินุกซ์แบบเรียลไทม์ (อาร์ทีลินุกซ์) ว่ามีคุณสมบัติเด่นๆอะไรบ้าง และมีวิธีการใช้งานอย่างไร

3) ได้รับความรู้จากการศึกษาภาษาซี ซึ่งเป็นภาษาที่ใช้ในการเขียนซอฟต์แวร์เพื่อทำงานร่วมกับระบบปฏิบัติการลินุกซ์แบบเรียลไทม์ (อาร์ทีลินุกซ์)

4) ได้เรียนรู้เกี่ยวกับการรับ-ส่งข้อมูลผ่านพอร์ตอนุกรม

5) ได้ศึกษาพฤติกรรมของการใช้งานระบบอาร์ทีลินุกซ์

1.5 ขั้นตอนการดำเนินงาน

1.5.1 ขั้นตอนการศึกษาทฤษฎี

เป็นขั้นตอนที่ทำการศึกษาทฤษฎีต่างๆ เช่น ความหมายของระบบแบบเรียลไทม์ซึ่งเป็นเทคโนโลยีที่ปัจจุบันนำมาใช้กันอย่างแพร่หลายในด้านการนำไปประยุกต์ใช้ควบคุมอุปกรณ์อิเล็คทรอนิคส์

1.5.2 ขั้นตอนการศึกษาซอฟต์แวร์

เป็นขั้นตอนในการศึกษาถึงซอฟต์แวร์ที่นำมาใช้ในการพัฒนาแบบเรียลไทม์คืออาร์ทีลินุกซ์

1.5.3 ขั้นตอนการเก็บรวบรวมข้อมูลต่างๆ

เป็นขั้นตอนที่นำเอาข้อมูลที่เกี่ยวข้องที่รวบรวม มาใช้ประกอบการศึกษา

1.5.4 ขั้นตอนการวิเคราะห์และออกแบบ

เป็นขั้นตอนที่จะทำการวิเคราะห์ระบบที่ศึกษา แล้วทำการเขียนโปรแกรมเพื่อรับส่งข้อมูลที่เป็นแบบเรียลไทม์

1.5.5 ขั้นตอนการตรวจสอบความถูกต้อง

เป็นขั้นตอนที่ทำการใช้โปรแกรมที่เขียนเพื่อทำการรับ-ส่งข้อมูลแล้วทำการเก็บผลเวลาที่ใช้ในการทำงานและวิเคราะห์ผลที่ได้รับจากการทดลอง

1.5.6 ขั้นตอนการทำเอกสารประกอบ

เป็นขั้นตอนที่สร้างเอกสารประกอบการใช้งานระบบระบบส่งผ่านข้อมูลที่เป็นแบบเรียลไทม์และเอกสารอ้างอิงในการศึกษาเพื่อทำปัญหาพิเศษ

1.6 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

1.6.1 รายละเอียดทางด้านอุปกรณ์คอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1) คอมพิวเตอร์ 2 เครื่อง

2) สายต่อพ่วงพอร์ทอนุกรม 1 เส้น

1.6.2 รายละเอียดทางด้านโปรแกรม

1) ระบบปฏิบัติการ Red Hat Linux 9.0

2) Kernel เวอร์ชั่น prepatched linux kernel 2.4.20-rtl

3) RTLinux Kernel เวอร์ชั่น rtlinux-3.1

4) gcc เวอร์ชั่น 3.2.2

5) rt_com เวอร์ชั่น 0.5.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

2.1 ระบบเรียลไทม์(Real Time System)

จากที่กล่าวมาว่าระบบเรียลไทม์มีอยู่ 2 ความหมายไม่ว่าจะเป็นการตอบสนองทันที หรือการทำงานให้เสร็จทันเวลาที่กำหนด ในทางทฤษฎี ระบบเรียลไทม์ ยังสามารถแบ่งตามชนิดของเดดไลน์ได้อีก

2.1.1 รูปแบบของเรียลไทม์

2.1.1.1 ฮาร์ดเรียลไทม์ (Hard Real Time)

หมายถึงระบบสำหรับงานที่ต้องการความเที่ยงตรงและความแม่นยำสูงในเรื่องของเวลา ระบบเรียลไทม์ที่เป็นฮาร์ดเรียลไทม์ ต้องสามารถรับประกันได้ว่าต้องทำงานเสร็จทันเดดไลน์ที่กำหนด และมีการจัดลำดับของงานทั้งหมดเป็นแบบสเตติกคือต้องรู้ลำดับความสำคัญของงานแต่ละงาน ตัวอย่างงานประเภทนี้ เช่น การยิงจรวด หากยิงไม่ตรงตามเวลาที่กำหนด ทำให้จรวดยิงไม่โดนเป้าหมาย หรือการควบคุมรถไฟฟ้า ซึ่งต้องทำการควบคุมความเร็วและรักษาระดับของการลดความเร็วเพื่อการคำนวณเวลาในการหยุด เพราะถ้าหากเงื่อนไขของเวลาผิดพลาดเกินกว่าที่ยอมรับได้อาจจะทำให้รถไฟฟ้าเกิดความเสียหายหรืออุบัติเหตุได้

2.1.1.2 ซอฟต์เรียลไทม์ (Soft Real Time)

หมายถึงระบบสำหรับงานที่ต้องการการทำงานที่รวดเร็วพอ แต่อาจมีความผิดพลาดในเรื่องของเวลาที่สามารถยอมรับได้ ซึ่งระบบยังคงทำตามลำดับของงานจนเสร็จ แม้จะผ่านเดดไลน์มาแล้วก็ตาม ซึ่งช่วงเวลาที่อนุโลมให้งานสามารถทำต่อได้นั้นเราเรียกว่า "เกรซพีเรียด(Grace Period)" ตัวอย่างของงานประเภทนี้เช่น การประมวลผลภาพเคลื่อนไหว เราต้องการให้มีความเร็วในการประมวลผลมากกว่า 30 เฟรมต่อวินาที การประมวลผลที่ช้าเกินไปจะทำให้ภาพที่ได้มีคุณภาพลดลง หรือการเล่นเพลงจะทำให้เกิดการสะดุด แต่เราก็สามารถยอมรับได้

2.2 ระบบปฏิบัติการแบบเรียลไทม์ (Real Time Operating System : RTOS)

ลักษณะโดยทั่วไปของระบบเรียลไทม์คือต้องคาดเดาได้(Predictably) และสามารถทำงานตามเวลาที่กำหนดได้(Schedulability)ในการนำคอมพิวเตอร์มาประยุกต์ใช้งานบางประเภทที่ต้องการมากกว่าการได้ผลลัพธ์ที่ถูกต้อง คือต้องทำงานได้ภายในระยะเวลาที่เรากำหนด หรือมีความผิดพลาดทางเวลาที่ยอมรับได้ อาจจะน้อยมากถึงขั้นเป็น1/1000 วินาที หรือน้อยกว่านั้น งานที่กล่าวถึงอย่างเช่น การควบคุมเครื่องจักรกล เราต้องอ่านข้อมูลสถานะของเครื่องจักรกลเพื่อใช้ในการคำนวณสัญญาณควบคุม และนำผลของสัญญาณที่คำนวณได้ไปควบคุมเครื่องจักรกลนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทั้งหมดนี้ต้องทำภายในระยะเวลาที่กำหนดแน่นอน ความผิดพลาดจากเวลาที่กำหนดอาจทำให้ไม่ได้สมรรถนะตามที่ต้องการ หรืออาจทำให้ระบบขาดเสถียรภาพและทำให้เกิดความเสียหายได้

จากคุณสมบัติของการทำงานแบบเรียลไทม์ส่วนใหญ่จะใช้ฮาร์ดแวร์และซอฟต์แวร์ที่ออกแบบมาโดยเฉพาะสำหรับงานนั้นๆ ทำให้การพัฒนาระบบและการนำไปใช้งานมีความยุ่งยากและอาจเสียค่าใช้จ่ายสูง นอกจากนั้นการเพิ่มความสามารถให้แก่ระบบที่มีอยู่เดิม เช่น การเพิ่มความสามารถในการเชื่อมต่อกับเครือข่าย การติดต่อกับผู้ใช้แบบกราฟิก หรือการติดต่อกับฐานข้อมูลจะทำได้ยากหรือไม่สามารถทำได้ ทั้งที่ความสามารถเหล่านี้มีอยู่ในระบบปฏิบัติการทั่วไปอยู่แล้ว

ดังนั้นจึงมีการพัฒนาระบบปฏิบัติการที่สามารถทำงานเรียลไทม์ได้และยังสามารถทำงานอื่นๆ ที่ระบบปฏิบัติการทั่วไปสามารถทำได้ รวมทั้งสามารถใช้งานบนเครื่องคอมพิวเตอร์ที่เป็นมาตรฐาน เช่น พีซี (PC : Personal Computer) ซึ่งมีใช้กันอยู่แพร่หลายและราคาถูก ปัจจุบันผลิตภัณฑ์ของระบบปฏิบัติการแบบเรียลไทม์มีอยู่มากมาย สำหรับตัวอย่างระบบปฏิบัติการแบบเรียลไทม์ที่นิยมใช้กันมากแสดงได้ดังตารางที่ 2.1

Operating System Name	Manufacture
pSOSystem 3	Wind River
QNX Neutrino RTOS	QNX Software System Ltd
VxWorks	Wind River
Windows CE.NET	Microsoft
RTLinux	FSMLabs
Real Time Application Interface (RTAI)	DIAPM

ตารางที่ 2.1 ผลิตภัณฑ์ของระบบปฏิบัติการแบบเรียลไทม์

ระบบปฏิบัติการแต่ละตัวจะมีจุดเด่นแตกต่างกันออกไป เช่น pSOSystem 3 จะสนับสนุนระบบแบบฝังตัว และงานทางด้านเครือข่ายที่ซับซ้อน QNX Neutrino RTOS ใช้กับงานทางด้านการแพทย์ , ระบบเกี่ยวกับการบิน และควบคุมอุปกรณ์ในรถยนต์ VxWorks ใช้ในการควบคุมหุ่นยนต์,ระบบจำลองการบิน และได้ขยายไปถึงระบบการติดต่อสื่อสาร Windows CE.NET จะเน้นทางด้านงานบนโทรศัพท์มือถือและกล้องถ่ายรูปดิจิตอล RTLinux ใช้ควบคุมเครื่องจักรในโรงงานอุตสาหกรรม , ควบคุมหุ่นยนต์ อาร์ทีเอไอ ใช้ควบคุมเครื่องจักรในโรงงานอุตสาหกรรม เป็นต้น

อาร์ทีลินุกซ์นั้นได้เริ่มขึ้นจาก University of New Mexico (UNM) ซึ่งผู้ที่พัฒนาคือ ดร.วิคเตอร์ โยไดเกน (Dr.Victor Yodaiken) และต่อมา FSMLabs ได้พัฒนาต่อจนถึงปัจจุบัน และในส่วนทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DIAPM ได้พัฒนา อาร์ทีเอไอขึ้นมาโดยมีพื้นฐานบนอาร์ทีลินุกซ์และว่ากันว่าอาร์ทีเอไอมีการใช้งานที่ง่ายกว่าอาร์ทีลินุกซ์[1] ซึ่งต่อมาได้มีการพัฒนาส่วนเพิ่มเติมของอาร์ทีเอไอโดยบริษัท DENX Software Engineering Pengutronix and Sysgo Realtime Solutions ในเชิงพาณิชย์ โดยที่ความแตกต่างระหว่างอาร์ทีลินุกซ์กับอาร์ทีเอไอแสดงได้ดังตารางที่ 2.2

	RTLinux	**RTAI**
Hardware	i386, PPC, ARM, Alpha, MIPS	i386, MIPS, PPC, ARM, m68k-nommu
Multi-processor	Yes	Yes
Scheduling	SCHED_FIFO, EDF, and RM	Fixed priority
Processes	Pthreads	Lightweight processes
Priorities lower – higher	(0-1,000,000)	(0x3fffFfff-0)
Memory protection	No	Yes
Dynamic memory	No	Yes
Inter-process communication	Semaphores, Mutexes, *Condition-var., FIFO*	Semaphores, Mutexes, Condition-var., FIFO, Mailbox, shared-mem, net_rpc, Pqueues.
Priority inversion control	Immediate ceiling	Inheritance
Time Resolution	Hardware dependant	Hardware dependant
Timers	None	None
Low level programming	Full control HW	Full control HW
Network	None	None
File Systems	None	None

ตารางที่ 2.2 ตารางแสดงความแตกต่างของระบบปฏิบัติอาร์ทีลินุกซ์ และอาร์ทีเอไอ

ตารางที่ 2.2 ได้ทำการเปรียบเทียบคุณสมบัติของระบบปฏิบัติการอาร์ทีลินุกซ์ และอาร์ทีเอไอ โดยที่ข้อแตกต่างหลักๆก็คือการจัดตารางการทำงานของโพรเซส(Scheduling) ซึ่งใช้ไม่เหมือนกัน และอีกข้อหนึ่งคือการทำงานของโพรเซสซึ่งอาร์ทีลินุกซ์จะเป็นพีเทรดเป็นมาตรฐานที่ถูกกำหนด

[1] Pierre Cloutier ได้นำเสนอผลงานชิ้นนอกว่า RTAI อยู่ ณ จุดใดในปัจจุบัน และทิศทางของมันในอนาคตเป็นอย่างไร โดย RTAI เป็นทางหนึ่งที่นิยมอย่างมากในการประยุกต์ใช้เรียลไทม์ลินุกซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดย POSIX[2](IEEE 1003.1c) ส่วนอาร์ทีเอไอจะเป็นไลท์เวทโพรเซส (การมีเทรดเป็นส่วนประกอบย่อยๆใน 1 โพรเซส หรือ มัลติเทรด) ซึ่งไม่ได้อ้างมาตรฐานของ POSIX

2.2.1 คุณสมบัติของระบบปฏิบัติการแบบเรียลไทม์

1) ขนาดเล็ก เพราะใช้กับระบบที่มีการใช้ทรัพยากรที่มีจำนวนจำกัด

2) เชื่อถือได้ ข้อมูลที่ผิดพลาดอาจเป็นผลเสียต่อระบบ ซึ่งระบบแบบเรียลไทม์ต้องทำให้มั่นใจได้ว่าสามารถทำงานได้อย่างถูกต้อง และรับประกันว่าต้องทำเสร็จทันเดดไลน์

3) เสถียรภาพ มีการปกป้องระบบกรณีระบบเกิดความเสียหายจะพยายามตรวจสอบและแก้ไข แล้วทำการรันงานต่อไป(เกิดการแฮงค์ได้ยาก)

4) มีความเร็ว การคอนเท็กซ์สวิตช์และสามารถตอบรับการอินเทอร์รัพท์จากภายนอกได้อย่างรวดเร็ว ยิ่งเร็วทำให้ตอบสนองได้ฉับพลัน เป็นผลมาจากการที่ตัวระบบไม่ซับซ้อน มีการแตกเทรดในการทำงาน และไม่มีหน่วยความจำสำรอง

5) สนับสนุนการทำงานแบบหลายๆงานพร้อมกัน(Multitasking)

6) ใช้ Preemptive Scheduling เป็นพื้นฐานในการจัดลำดับความสำคัญของงาน เมื่อมีงานที่เป็นเรียลไทม์เข้ามาจะให้ความสำคัญมากกว่างานที่ไม่ใช่เรียล์ไทม์

2.3 ลินุกซ์ (Linux)

ลินุกซ์เป็นระบบปฏิบัติการที่พัฒนามาจากระบบยูนิกซ์โดยไลนัส ทอร์วัลดส์ (Linus Torvalds) ลินุกซ์ถูกเผยแพร่ทางระบบอินเตอร์เน็ต และให้ทุกคนที่มีความสนใจได้ดาวน์โหลดไปทดลองใช้งาน ทำให้มีโปรแกรมเมอร์มากมายมาช่วยแก้ไขปรับปรุงและพัฒนาลินุกซ์

ในความหมายด้านเทคนิคลินุกซ์เป็นเพียงเคอร์เนลของระบบปฏิบัติการ ซึ่งจะทำหน้าที่ในด้านจัดสรรและบริหารทรัพยากรของระบบปฏิบัติการ จัดการไฟล์และการติดต่อกับอุปกรณ์ต่างๆ แต่ผู้ใช้ทั่วๆไปจะรู้จักลินุกซ์ผ่านระบบอินเทอร์เฟสที่พวกเขาเห็น เช่น เชลล์ (Shell) หรือเอ็กซ์วินโดว์ (XWindow)

เนื่องจากลินุกซ์ได้พัฒนาภายใต้ลิขสิทธิ์แบบ GPL (GNU General Public License) ซึ่งเป็นลิขสิทธิ์ที่ยอมให้มีการเปลี่ยนแปลงต้นฉบับหรือแจกจ่ายได้โดยไม่จำกัดสิทธิ์จึงสามารถดาวน์โหลดฟรีได้จากอินเตอร์เน็ต

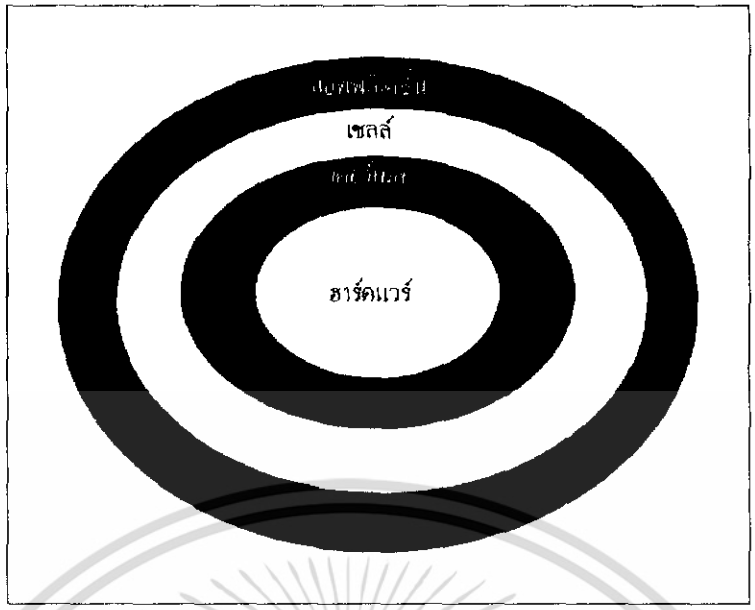
2.3.1 โครงสร้างพื้นฐานในการทำงานของลินุกซ์

ลินุกซ์มีโครงสร้างพื้นฐานในการทำงาน 4 ส่วนด้วยกันโดยจะแสดงให้เห็นดังรูปที่ 2.1

---

[2] POSIX เป็นมาตรฐานของ Source code level ซึ่งเป็นมาตรฐานนี้จะเอื้ออำนวยในส่วนของ Interface เพื่อที่จะให้ Application เรียกใช้งานเช่น คำสั่งจำพวก System Call โดยที่ POSIX ถูกพัฒนาโดย IEEE และรับรองมาตรฐานโดย ANSI และ ISO ซึ่ง POSIX จะเป็นมาตรฐานบนระบบ Unix [5]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 โครงสร้างพื้นฐานในการทำงานของลินุกซ์

1) ฮาร์ดแวร์ (Hardware) หมายถึง ตัวเครื่องคอมพิวเตอร์ ส่วนประกอบต่างๆ เช่น ซีพียู เมนบอร์ด จอภาพ คีย์บอร์ด เมาส์ ฮาร์ดิสก์ หน่วยความจำ ฯลฯ

2) เคอร์เนล (Kernel) หมายถึง ส่วนที่ใช้ควบคุมการทำงานของทั้งระบบทั้งหมด มีหน้าที่และบริหารการใช้หน่วยความจำ ตรวจสอบและคอยควบคุมอุปกรณ์ฮาร์ดแวร์

3) เชลล์ (Shell) หมายถึง ชั้นหรือตัวกลางที่ใช้เชื่อมต่อระหว่างเคอร์เนลกับยูสเซอร์ หรือยูสเซอร์กับทรัพยากรต่างๆของระบบ เชลล์จะคอยรับคำสั่งต่างๆ จากยูสเซอร์ผ่านทางคีบอร์ด แล้วทำการแปลคำสั่งให้เป็นภาษาเครื่อง

4) แอพพลิเคชั่น (Application) หมายถึง โปรแกรมประยุกต์ต่างๆที่เขียนขึ้นมาใช้งาน

2.3.2 หลักการที่ใช้ในการออกแบบลินุกซ์

สิ่งที่ใช้พิจารณาในการออกแบบลินุกซ์เคอร์เนลมีดังต่อไปนี้

1) ความชัดเจน (Clarity) คือ ลินุกซ์เคอร์เนลที่ออกแบบขึ้นมาต้องมีความชัดเจน สามารถทำความเข้าใจได้ง่ายทั้งส่วนเรื่องของความเร็วและความทนทาน หมายความว่าสามารถพิสูจน์ความถูกต้องของข้อมูลได้ง่าย และเมื่อเกิดความผิดพลาดขึ้นสามารถทำการดีบักได้ง่าย ทำให้ระบบเกิดความเสียหายได้ยาก

2) ความเข้ากันได้ (Compatibility) ลินุกซ์เคอร์เนลสามารถนำไปใช้งานร่วมกับยูนิกซ์ได้โดยสนับสนุนมาตรฐาน POSIX (Portable Operating System Interface) และยังสามารถใช้งานร่วมกับตัวอื่นๆ ได้แก่ สนับสนุนการรันไฟล์ .class ของจาวา สามารถทำการเอ็กซิคิวท์ DOS ผ่าน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัว DOSEMU ทำการแชร์ไฟล์ได้เหมือนกับวินโดว์ และสามารถใช้งานไดรฟ์จากระบบปฏิบัติการอื่นๆได้รวมทั้งสนับสนุนระบบเน็ตเวิร์กด้วยโปรโตคอล TCP/IP

3) สะดวกในการโยกย้าย (Portable) ลินุกซ์เคอร์เนลสามารถใช้งานร่วมกับฮาร์ดแวร์หลายแพลตฟอร์มได้ เนื่องจากโคดของลินุกซ์เคอร์เนลไม่ยึดติดกับสถาปัตยกรรม ทำให้นำไปใช้งานได้หลากหลาย

4) ทนทานและมีความปลอดภัย (Robustness and Security) ลินุกซ์เคอร์เนลที่พัฒนาขึ้นต้องมีความทนทานและปลอดภัย ควรจะมีการป้องกันระบบจากระบบอื่น โดยเฉพาะทำการป้องกันโพรเซสที่ทำงานจากส่วนอื่นๆ สามารถทำการดีบักและแก้ไขข้อผิดพลาดได้ง่าย

5) เร็ว (Speed) เป็นส่วนที่ต้องพิจารณาอันดับแรกซึ่งมีความสำคัญมาก ประสิทธิภาพการทำงานของระบบสามารถตรวจสอบได้จากความเร็วในการรันงานต่างๆ

2.3.3 ข้อดีของลินุกซ์

1) สนับสนุนการทำงานหลายๆงานพร้อมๆกัน (Multitasking)

2) ยอมให้ผู้ใช้สามารถเข้ามาทำงานพร้อมๆกันได้ (Multi-user)

3) เนื่องจากภาษาที่ใช้พัฒนาคือภาษาซีซึ่งทำให้ทำงานร่วมกับฮาร์ดแวร์ได้เร็ว

4) มีเสถียรภาพสูง

5) ทำงานได้บนหลายสถาปัตยกรรม เช่น X86, Power PC, Alpha, Hitachi SH, และ Intel Strong ARM

6) เป็นระบบปฏิบัติการที่หาดาวน์โหลดฟรีได้จากอินเตอร์เน็ต

2.4 อาร์ทีลินุกซ์ (RTLinux)

ลินุกซ์เป็นระบบปฏิบัติการที่สามารถใช้ได้โดยไม่ต้องเสียค่าใช้จ่าย ดังนั้นจึงมีคนได้นำลินุกซ์มาประยุกต์ใช้กับการทำงานแบบเรียลไทม์ แต่พบว่าโดยตัวของมันเองไม่เหมาะที่จะนำมาใช้ในการทำงานแบบเรียลไทม์ ดังนั้นจึงได้มีการพัฒนาระบบปฏิบัติการอาร์ทีลินุกซ์ขึ้นโดยกลุ่มนักวิจัย ที่ New Mexico Institute of Technology ซึ่งมี ดร.วิคเตอร์ โยไดเกน (Dr. Victor Yodaiken) เป็นหัวหน้าโครงการ การพัฒนาได้เริ่มมาตั้งแต่ปี ค.ศ. 1996 ในขณะนี้ได้พัฒนาถึงรุ่นที่ 3.2-pre3

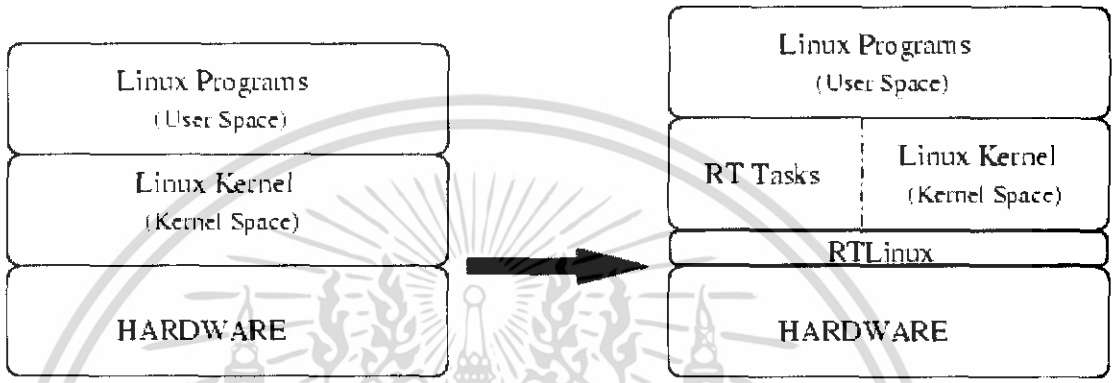
อาร์ทีลินุกซ์เป็นระบบปฏิบัติการที่รวมเอาความสามารถของระบบปฏิบัติการทั่วไป (General-purpose OS) ซึ่งก็คือ ลินุกซ์ เข้ากับความสามารถของการทำงานแบบเรียลไทม์ทำให้เราสามารถใช้อาร์ทีลินุกซ์ในการทำงานที่ต้องการความเที่ยงตรงของเวลาตอบสนอง เช่นในการเก็บตัวอย่างข้อมูลหรือการควบคุมวงจรปิดโดยใช้คอมพิวเตอร์และยังสามารถใช้ความสามารถของลินุกซ์ เช่น ความสามารถในด้านเครือข่าย (networking) หรือ การติดต่อกับผู้ใช้แบบกราฟิก (GUI) ได้อีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เช่นเดียวกับลินุกซ์ อาร์ทีลินุกซ์เป็นซอฟแวร์ที่ใช้ลิขสิทธ์แบบ GPL ผู้ใช้สามารถนำโค้ดไปพัฒนาต่อให้เหมาะสมกับการใช้งานยิ่งขึ้นได้ และผู้ใช้ไม่ต้องเสียค่าใช้จ่ายให้ผู้ทำอาร์ทีลินุกซ์แต่อย่างใด

2.4.1 โครงสร้างการทำงานของของอาร์ทีลินุกซ์

การที่จะทำให้ลินุกซ์มีความสามารถทำงานในแบบเรียลไทม์ ทำได้โดยการเพิ่มชั้นของซอฟต์แวร์ระหว่างฮาร์ดแวร์ของคอมพิวเตอร์กับลินุกซ์เคอร์เนลดังรูปที่ 2.2



รูปที่ 2.2 การเพิ่มชั้นของอาร์ทีลินุกซ์เข้าไป

โดยอาร์ทีลินุกซ์จะมองงานในลินุกซ์เคอร์เนลเป็นงานที่ความสำคัญต่ำ จะทำงานได้ก็ต่อเมื่อไม่มีงานแบบเรียลไทม์ (RT Tasks) ในคิวเท่านั้น และเมื่องานแบบเรียลไทม์ต้องการทำงาน อาร์ทีลินุกซ์จะสามารถหยุดการทำงานของลินุกซ์เคอร์เนลได้ตลอดเวลาไม่ว่างานของลินุกซ์จะทำอะไรอยู่ก็ตาม ส่วนตัวลินุกซ์เองจะไม่เห็นความแตกต่างไปจากเดิม จึงสามารถใช้งานทุกอย่างไม่ว่าจะเป็นความสามารถด้านเครือข่ายหรือด้านกราฟิกได้เหมือนเดิม

ในกรณีที่เกิดอินเทอร์รัพท์จากฮาร์ดแวร์ อาร์ทีลินุกซ์จะทำการดักสัญญาณอินเทอร์รัพท์นั้นไว้ ถ้าไม่มีงานเรียลไทม์อื่นๆต้องการสัญญาณที่ดักได้นี้จึงจะส่งต่อไปให้ลินุกซ์ และกรณีที่ลินุกซ์ทำการดิสเอเบิลอินเทอร์รัพท์เพื่อเหตุผลใดก็ตาม อาร์ทีลินุกซ์จะไม่ทำการดิสเอเบิลอินเทอร์รัพท์นั้นจริงๆแต่จะไม่ส่งสัญญาณอินเทอร์รัพท์นั้นให้กับลินุกซ์เคอร์เนลเท่านั้น

เพื่อความรวดเร็วในการทำงาน อาร์ทีลินุกซ์จึงเป็นชั้นของซอฟต์แวร์ที่มีขนาดเล็กใช้ตารางการจัดลำดับงานแบบง่ายๆ โดยจะให้งานที่มีความสำคัญมากกว่าได้ทำงานก่อนเสมอ และงานเรียลไทม์จะไม่มีการถูกสลับลงฮาร์ดดิสก์ อีกทั้งงานในเวลาจริงทั้งหมดจะใช้พื้นที่ของหน่วยความจำเดียวกันกับลินุกซ์เคอร์เนล ซึ่งเราเรียกว่าการทำงานในเคอร์เนลสเปซ(kernel space) จึงทำให้ไม่เสียเวลามากในการหยุดการ ทำงานของงานหนึ่งและให้อีกงานหนึ่งทำต่อ (fast context switch time) จาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เหตุผลดังที่กล่าวมาทำให้อาร์ทีลินุกซ์สามารถให้ความเที่ยงตรงในระดับ $1/10^6$วินาที(Microsecond) บนเครื่องคอมพิวเตอร์ระดับเพ็นเทียมคลาสได้ [3]

2.4.2 ข้อดีเพิ่มเติมของอาร์ทีลินุกซ์

1) อาร์ทีลินุกซ์ ได้ทำให้สามารถทำงานแบบฮาร์ดเรียลไทม์ได้

2) อาร์ทีลินุกซ์ สามารถใช้เครื่องคอมพิวเตอร์ได้เหมือนกับระบบแบบฝังตัว (Embedded System) ซึ่งจะมีความได้เปรียบในเรื่องของทรัพยากรสำหรับการพัฒนาและการแก้ไขซอฟต์แวร์

2.5 การติดต่อแลกเปลี่ยนข้อมูลระหว่างกระบวนการของอาร์ทีลินุกซ์(RTLinux Interprocess Communication Mechanism)

อาร์ทีลินุกซ์ได้รับการออกแบบให้มีขนาดเล็กและไม่ซับซ้อน ดังนั้นงานแบบเรียลไทม์จึงไม่สามารถทำงานที่ต้องการการบริการของลินุกซ์ได้เช่น การติดต่อกับผู้ใช้งาน, การอ่านเขียนข้อมูลในฮาร์ดดิสก์ หรือการติดต่อกับเครือข่าย งานดังกล่าวจะต้องทำโดยโปรแกรมที่ทำงานในลินุกซ์

อาร์ทีลินุกซ์จึงได้จัดช่องทางที่ทำให้งานเรียลไทม์สามารถติดต่อกับโปรแกรมที่ทำงานในลินุกซ์ไว้ 2 วิธี คือการติดต่อผ่านทาง FIFO ซึ่งมีลักษณะคล้ายกับการติดต่ออ่านเขียนข้อมูล และอีกช่องทางคือติดต่อผ่านทาง Shared Memory[4] ในโครงงานนี้ได้เลือกการใช้งาน FIFO สำหรับการรับส่งข้อมูลระหว่างโมดูลกับโปรแกรมประยุกต์ ทั้งนี้เพราะ FIFO มีการใช้งานที่ง่าย ไม่ยุ่งยากซับซ้อน

---

[3] Phil Wilshire ได้ทำการทดสอบและประเมินผลส่วน Suite Toolkit เพื่อวัดประสิทธิภาพของ RTLinux ว่ามันดีแค่ไหน ซึ่งผลลัพธ์ก็ยอดเยี่ยมแม้จะใช้ฮาร์ดแวร์แบบธรรมดา ๆ [7]

[4] สำหรับการใช้ Shared Memory สามารถใช้ได้ผ่าน mbuff โดยมี Tomasz Motylewski เป็นผู้พัฒนา [7] ซึ่งมันถูกบรรจุอยู่ใน RTLinux เมื่อเวลาทำการ Run และมันได้ถูกติดตั้งใน "drivers/mbuff"

mbuff.o เป็นโมดูลที่ต้องแทรกเข้าไปใน kernel space ซึ่งโมดูลนี้จะมี 2 ฟังก์ชันเพื่อทำการจองบล็อกในการจะทำการใช้ Shared Memory ดังนี้

```
#include <mbuff.h>
void * mbuff_alloc(const char *name, int size)
void mbuff_free(const char *name, void * mbuf)
```

ฟังก์ชันแรกจะเป็นการจองเลือกหน่วยความจำที่ต้องการใช้โดยจะเรียกผ่านชื่อ name และ ขนาดเท่ากับ size ถ้าสามารถจองเลือกหน่วยความจำได้ name จะคืนค่าเป็น address ของหน่วยความจำ และถ้าจองไม่ได้จะคืนค่าเป็น NULL และถ้ามีการจองบล็อกหน่วยความจำที่ชื่อ name นี้แล้ว จะคืนค่ามาเป็น address ของบล็อกหน่วยความจำชื่อ name ที่มีอยู่แล้ว และในส่วนฟังก์ชันที่ 2 จะเป็นการคืนทรัพยากรให้แก่ระบบ

"mbuff_alloc และ mbuff_free ไม่สามารถใช้ใน real time thread ได้ จะต้องเรียกผ่าน init_module และ cleanup_module เท่านั้น"[26]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

FIFO(First In First Out) หรือ เข้าก่อนออกก่อนคือวิธีการรับส่งข้อมูลระหว่างกระบวนการวิธีหนึ่ง เป็นชื่อที่ได้มาจากลักษณะการถ่ายเทข้อมูล โดยข้อมูลที่ส่งเข้าไปใน FIFO ก่อนหากมีการเรียกจะถูกเรียกออกมาก่อน ดังรูปที่ 2.3 ในอาร์ทีลินุกซ์ได้จัดเตรียมวิธีการสร้าง FIFOให้ การสร้างจะต้องสร้างด้วยโมดูล การอ่านและเขียนลงไปใน FIFO จึงจะทำได้ทั้งจากโมดูลและโปรแกรมประยุกต์ โดยปกติเราจะสามารถสร้าง FIFO ได้ทั้งหมด 64 ชุด(เราสามารถเพิ่มจำนวนชุดได้จากการปรับตัวเลือกของอาร์ทีลินุกซ์ในตัวเลือกที่เกี่ยวกับ Driver) ขนาดหน่วยความจำที่จะใช้สร้าง FIFO แต่ละชุดรวมกันแล้วจะถูกจำกัดโดยขนาดหน่วยความจำของเครื่องคอมพิวเตอร์

รูปที่ 2.3 การทำงานของ FIFO

การอ่านหรือเขียนข้อมูลของ FIFO ในโมดูลจะใช้ฟังก์ชันเฉพาะแต่ในโปรแกรมประยุกต์จะมีการติดต่อกับ *FIFO* เหมือนกับการติดต่ออ่านหรือเขียนแฟ้มข้อมูลทั่วไปผ่านแฟ้มข้อมูลชื่อ "/dev/rtf{fifo}" (เมื่อ fifo มีค่าได้ตั้งแต่ 0 ถึง 63) ที่ถูกสร้างขึ้นจากขั้นตอนการติดตั้งอาร์ทีลินุกซ์ นอกจาก FIFO จะเป็นทางผ่านของข้อมูลแล้วอาร์ทีลินุกซ์ยังได้เพิ่มความสามารถในการเรียกใช้งานฟังก์ชันที่กำหนดขึ้นหากมีการอ่าน หรือเขียน FIFO ด้วยโปรแกรมประยุกต์หรือโมดูล ความสามารถนี้เราจะนำไปใช้ในการสั่งงานโมดูลเวลาจริงให้เริ่มหรือหยุดการทำงาน

2.6 วิธีการจัดลำดับตารางการทำงาน (Scheduling Characteristic)

การจัดตารางการทำงานเป็นหน้าที่พื้นฐานของระบบปฏิบัติการ เนื่องจากระบบปฏิบัติการต้องคอยจัดสรรทรัพยากรให้กับโพรเซสเข้าใช้งาน

2.6.1 การจัดตารางการทำงานแบบมาก่อนได้ก่อน(First In First Out:FIFO)

เป็นวิธีการจัดการทำงานแบบที่ง่ายที่สุดคือโพรเซสใดมาขอเข้าใช้ซีพียูก่อนก็จะได้ทำงานก่อน ส่วนโพรเซสอื่นที่เข้ามาในขณะที่ซีพียูยังไม่ว่างก็จะรออยู่ในคิวจนกว่าซีพียูว่างจึงได้รับให้เข้าทำงาน

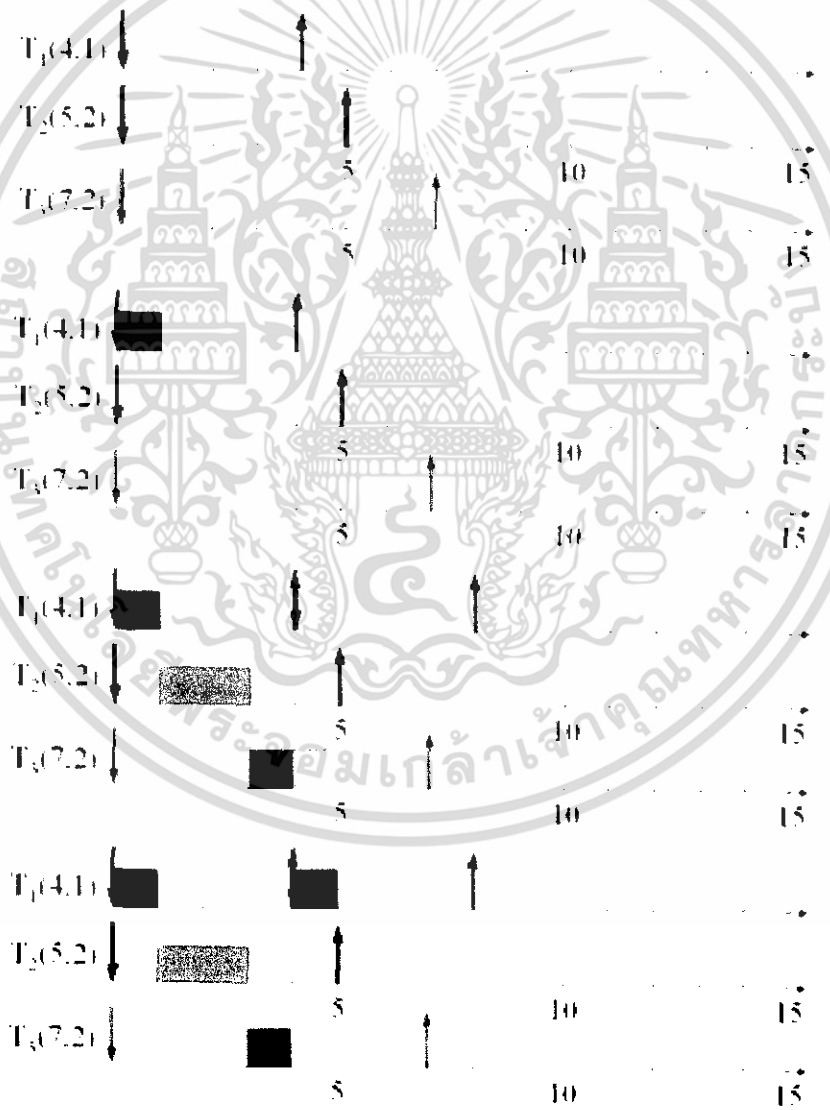
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.2 การจัดตารางการทำงานแบบเวียนเทียน (Round Robin:RR)

การจัดตารางการทำงานโดยวิธีนี้จะมีการจัดส่วนแบ่งเวลาการทำงานของโพรเซส หมายความว่า จะไม่มีโพรเซสไหนได้ครอบครองซีพียูเกินส่วนแบ่งเวลาที่กำหนดไว้ ถ้าเกินโพรเซสนั้นก็จะเข้าไปต่อในคิวและรอการกลับเข้าใช้ซีพียูใหม่อีกครั้ง

2.6.3 การจัดตารางแบบ Rate Monotonic (RM)

เป็นการจัดตารางที่ใช้ในระบบปฏิบัติการแบบเรียลไทม์ หลักการของการจัดตารางรูปแบบนี้เป็นแบบสเตติกไพรออริตี หมายความว่ามันจะจัดคิวการทำงานโดยให้ไพรออริตีสูงกับงานที่มีคาบเวลาการทำงานน้อยที่สุดก่อน แสดงได้ดังรูปที่ 2.4 ฟังก์ชัน T(p,e) คืองานในคิวโดยที่ p คือ คาบเวลาการทำงาน(Period time) และ e คือ เวลาที่ใช้ในการทำงาน (Execute time)

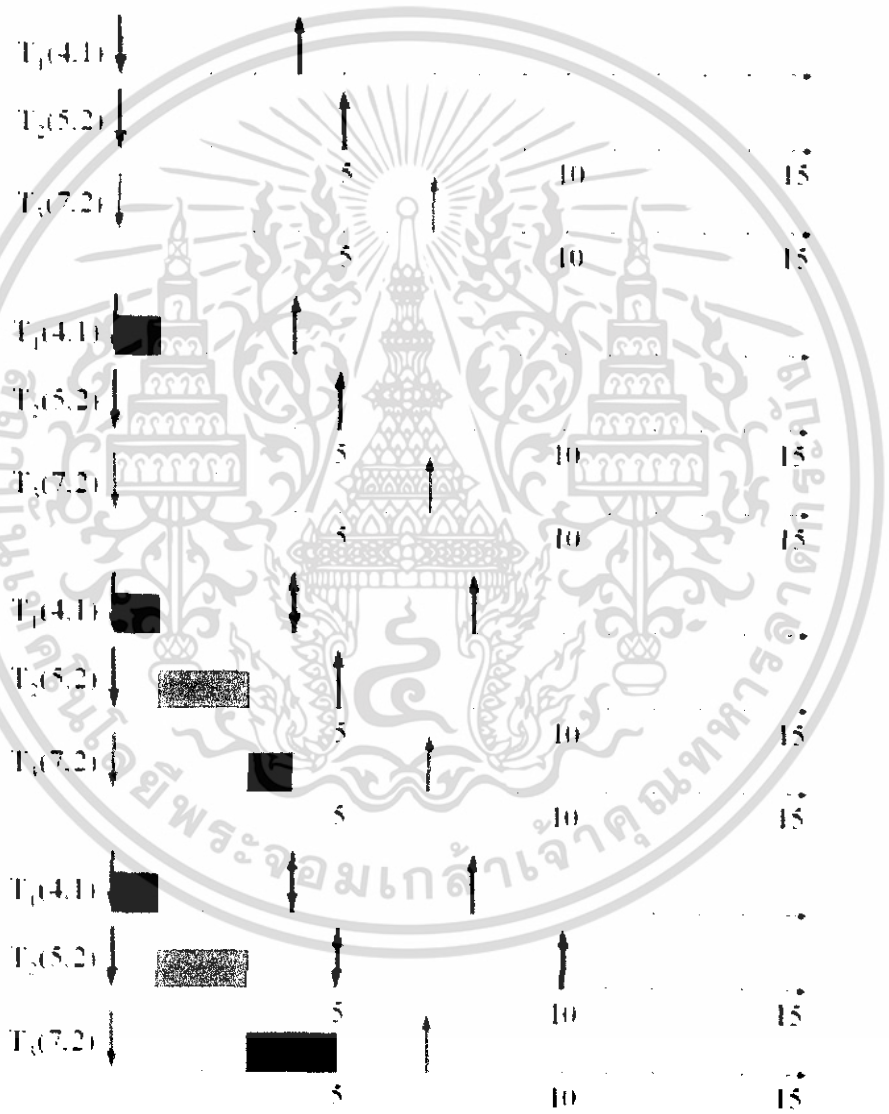


รูปที่ 2.4 การทำงานของงานที่ใช้การจัดตารางแบบ RM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.4 การจัดตารางแบบ *Earliest Deadline First* (EDF)

เป็นหลักการจัดตารางการทำงานที่ใช้ในระบบปฏิบัติการแบบเรียลไทม์ โดยการจัดลำดับการทำงานเป็นแบบไดนามิกไพรออริตี กล่าวคือ จะกำหนดไพรออริตีสูงให้กับงานที่มีคาบเวลาน้อยสุด เหมือนกับวิธีของ RM แต่ดิวการทำงานจะนำเดดไลน์เข้ามาพิจารณาด้วย โดยจะให้สิทธ์กับงานที่ใกล้จะถึงเดดไลน์มากที่สุดทำก่อนแต่ถ้าเหลือเวลาที่ใกล้เดดไลน์เท่ากันก็จะพิจารณาจากคาบเวลาในการทำงานที่น้อยที่สุดให้เข้ามาทำก่อน ถ้าต้องการใช้การจัดลำดับการทำงานแบบ EDF นี้สามารถหาได้จากเว็บไซต์[28] การจัดลำดับการทำงานแบบ EDF สามารถแสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 การทำงานของงานที่ใช้การจัดตารางแบบ EDF

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ขั้นตอนและวิธีการดำเนินงาน

ในการทำปัญหาพิเศษเรื่อง "การศึกษาการใช้งานของระบบเรียลไทม์ผ่านการติดต่อแบบอนุกรม" มีขั้นตอนในการดำเนินงานดังนี้

1) ศึกษาค้นคว้าและทำความเข้าใจเกี่ยวกับระบบปฏิบัติการแบบเรียลไทม์บนพื้นฐานของระบบปฏิบัติการลินุกซ์

2) ศึกษาและทำการติดตั้งระบบปฏิบัติการแบบเรียลไทม์(อาร์ทีลินุกซ์)

3) ศึกษาการเขียนโปรแกรมเพื่อทำการติดต่อกับอุปกรณ์ผ่านทางพอร์ตอนุกรมโดยใช้ภาษาซี

4) กำหนดขอบเขตที่จะทำสร้างระบบส่งผ่านข้อมูลที่เป็นแบบเรียลไทม์

5) สร้างระบบส่งผ่านข้อมูลที่เป็นแบบเรียลไทม์

6) วิเคราะห์ความถูกต้องของระบบที่สร้าง

7) ทำการสรุปผล

8) จัดทำเอกสาร

โดยที่เนื้อหาในบทที่ 3 นี้จะเป็นการแสดงรายละเอียดการดำเนินการใน 4 ข้อแรก นั่นคือ ศึกษาหลักการทำงานของระบบปฏิบัติการแบบเรียลไทม์ ศึกษาหลักการทำงานและการติดตั้งอาร์ทีลินุกซ์ ศึกษาการเขียนโปรแกรมเพื่อทำการติดต่อกับอุปกรณ์ผ่านทางพอร์ตอนุกรมโดยใช้ภาษาซี กำหนดขอบเขตที่จะทำสร้างระบบส่งผ่านข้อมูลที่เป็นแบบเรียลไทม์

3.1 ศึกษาค้นคว้าและทำความเข้าใจเกี่ยวกับระบบปฏิบัติการแบบเรียลไทม์บนพื้นฐานของระบบปฏิบัติการลินุกซ์

ในการศึกษาหลักการทำงานของระบบปฏิบัติการแบบเรียลไทม์ที่อยู่บนพื้นฐานของระบบปฏิบัติการลินุกซ์ เพื่อที่จะให้เข้าใจหลักการของการทำงาน และ ทราบถึงปัญหาที่ทำให้มีความจำเป็นที่ต้องใช้ความสามารถของระบบปฏิบัติการที่เป็นแบบทำงานในเรียลไทม์ โดยในการศึกษาได้ศึกษาค้นหาโดยเจาะจงตัวแก่นของระบบปฏิบัติการคืออาร์ทีลินุกซ์จากทั้งทางอินเตอร์เน็ต และจากทางหนังสือ ซึ่งมีทั้งที่เป็นเอกสาร, บทความ และ หนังสือต่างประเทศ เนื้อหาที่ทำการศึกษามีดังนี้

1) ระบบปฏิบัติการแบบเรียลไทม์ คืออะไร

2) คุณสมบัติของระบบปฏิบัติการแบบเรียลไทม์

3) หลักการทำงานของระบบปฏิบัติการแบบเรียลไทม์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง จุฬาลงกรณ์มหาวิทยาลัย

จากการศึกษารูปแบบ คุณสมบัติของมันแล้วจะทำให้สามารถมองเห็นถึงความสามารถที่จะนำไปประยุกต์ใช้ในการออกแบบและการพัฒนาระบบอื่น ๆ ต่อไป

3.2 ศึกษาและทำการติดตั้งระบบปฏิบัติการแบบเรียลไทม์ (RTLinux)

ในการศึกษาการติดตั้งของระบบปฏิบัติการแบบเรียลไทม์ เพื่อที่จะให้สามารถแก้ปัญหาที่เกิดขึ้นขณะที่ทำการติดตั้งได้ โดยในการศึกษาได้ค้นคว้าจากทางอินเตอร์เน็ต และ หนังสือซึ่งมีทั้งที่เป็นเอกสาร, บทความ และหนังสือต่างประเทศ

จากการศึกษาจะทำให้ขั้นตอนการติดตั้งระบบปฏิบัติการลินุกซ์ และการแก้ไขเคอร์เนลให้สามารถทำงานเป็นแบบเรียลไทม์ เป็นไปได้อย่างง่ายขึ้น

3.2.1 การติดตั้งระบบปฏิบัติการอาร์ทีลินุกซ์

การติดตั้งระบบปฏิบัติการอาร์ทีลินุกซ์เป็นการเปลี่ยนเคอร์เนลของระบบปฏิบัติการลินุกซ์เดิม ซึ่งอาจจะเป็นของผู้เผยแพร่รายใดก็ได้เช่น RedHat, Mandrake, Debian หรือ TLE มาใช้เคอร์เนลของของอาร์ทีลินุกซ์ (RT-Kernel) ที่พร้อมจะแทรกโมดูลการทำงานแบบเรียลไทม์ได้ในขณะที่โปรแกรมการทำงานในส่วนต่างๆ ของลินุกซ์ยังคงเหมือนเดิม

อาร์ทีลินุกซ์ที่เผยแพร่ในอินเตอร์เน็ตนั้นจะอยู่ในรูปของซอร์สโคด การติดตั้งจำเป็นที่เราจะต้องปรับค่าตัวเลือกและคอมไพล์ซอร์สโคดต่างๆเอง ดังนั้นผู้ติดตั้งควรจะมีความรู้เรื่องการคอมไพล์เคอร์เนลอยู่บ้าง การติดตั้งอาร์ทีลินุกซ์ประกอบด้วย 3 ขั้นตอนใหญ่ ๆ ด้วยกัน ขั้นแรก คือการติดตั้งระบบปฏิบัติการ โดยที่จะเป็นของผู้เผยแพร่รายใดก็ได้แล้วแต่ความชอบใจของผู้ใช้งาน ในที่นี้เราเลือกใช้ RedHat 9 ในขั้นตอนนี้อยู่นอกเหนือขอบเขตของโครงงานอีกทั้งเอกสารที่อธิบายเกี่ยวกับวิธีการติดตั้งก็มีอยู่ทั่วไป จึงจะขอไม่กล่าวถึง ขั้นที่สอง คือการติดตั้งเคอร์เนลของอาร์ทีลินุกซ์ และขั้นสุดท้าย คือการติดตั้งโมดูลที่จะใช้สำหรับการทำงานแบบเรียลไทม์ โดยขั้นตอนการติดตั้งเคอร์เนลของอาร์ทีลินุกซ์สามารทำได้ 2 วิธีหลัก ๆ คือ ทำการ patch ทับบนเคอร์เนลของลินุกซ์เดิมหรือทำการติดตั้งลงบนเคอร์เนลใหม่ที่ได้ทำการดาวน์โหลดมาซึ่งขอแนะนำวิธีที่ 2 เพราะวิธีที่ 1 อาจมีปัญหาในกรณีที่การปรับค่าเคอร์เนลที่ไม่ตรงกับลักษณะของฮาร์ดแวร์อาจส่งผลให้บูตเคอร์เนลนั้นไม่ได้แต่ถ้าเราลงที่คอร์เนลใหม่แล้วบูตไม่ขึ้นก็สามารถบูตเคอร์เนลเก่าได้

ขั้นตอนการติดตั้งโดยสังเขป

1) ทำการติดตั้งระบบปฏิบัติการลินุกซ์

2) แตกไฟล์ของลินุกซ์เคอร์เนล[1]ตามบรรทัดที่ 1และทำการสร้างลิงค์ตามบรรทัดที่ 2 ตามภาคผนวก ค.

---

[1] เว็บไซท์ที่สามารถดาวน์โหลดไฟล์ที่เกี่ยวข้องในการติดตั้งอาร์ทีลินุกซ์

- www.fsmlabs.com/products/download.htm
- คู่มือแนะนำการติดตั้งระบบปฏิบัติการอาร์ทีลินุกซ์ [4]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

59394

3) แตกไฟล์อาร์ทีลินุกซ์ตามบรรทัดที่ 3 และทำการ patch เคอร์เนลที่เป็นอาร์ทีลินุกซ์ตามบรรทัดที่ 5 ในภาคผนวก ค.

4) ปรับค่าลินุกซ์เคอร์เนลตามบรรทัดที่ 6 และคอมไพล์ตามบรรทัดที่ 7 ถึงบรรทัดที่ 10 ในภาคผนวก ค.

5) สำเนาไฟล์สำหรับบูตไปแทนที่ตัวเก่าให้ทำตามบรรทัดที่ 11 ในภาคผนวก ค.

6) ทำการแก้ไขไฟล์ของบูตโหลดเดอร์เพื่อให้สามารถบูตเข้าอาร์ทีเคอร์เนลได้ทำตามบรรทัดที่12-15 ในภาคผนวก ค.

7) รีบูตเคอร์เนลที่เป็นเรียลไทม์

8) สร้างลิงค์ตามบรรทัดที่ 17 ในภาคผนวก ค.

9) ปรับค่าของอาร์ทีลินุกซ์เคอร์เนลใช้คำสั่งตามบรรทัดที่ 18 ในภาคผนวก ค.

10) คอมไพล์เคอร์เนล ตามบรรทัดที่ 19-20 และ ทำการทดสอบโมดูลตามบรรทัดที่ 21 ในภาคผนวก ค.

ในส่วนของการติดตั้งระบบอาร์ทีลินุกซ์โดยละเอียดสามารถดูได้จากภาคผนวก ค.

3.3 ศึกษาสิ่งที่เกี่ยวข้องกับการรับและส่งข้อมูลผ่านพอร์ตอนุกรม

จากการที่จะต้องสร้างระบบส่งผ่านข้อมูลที่เป็นแบบเรียลไทม์โดยมีพื้นฐานอยู่บนระบบปฏิบัติการลินุกซ์แล้ว ซึ่งจะต้องมีการเขียนโปรแกรมเพื่อที่จะรับและส่งข้อมูลผ่านทางพอร์ตอนุกรมนั้นทางผู้จัดทำได้ใช้โมดูลที่เป็นมาตรฐานของการติดต่อสื่อสารของอาร์ทีลินุกซ์ที่มีชื่อว่า rt_com[2] โดยในโมดูลนี้ได้มีฟังก์ชันให้เลือกใช้ซึ่งทำให้ง่ายสำหรับการเขียนโปรแกรมเพื่อ

---

[2] นอกจากที่สามารถใช้โมดูล rt_com ในการรับส่งข้อมูลผ่านพอร์ตอนุกรมแล้วยังมี API ที่สามารถใช้แทนโมดูล rt_com นี้ได้ โดยที่ถ้าต้องการใช้ API นี้จะต้องทำการแทรกโมดูล rtl_posixio.o เข้าไปใน kernel space ซึ่งโปรแกรมนี้จะเปิดการติดต่อกับ /dev/mem, จากนั้นจึงทำการอ่านหรือเขียนข้อมูลต่อไปโดยเราสามารถดูได้ mapped area

นอกจากนั้นยังมีการรับส่งข้อมูลผ่านทางพอร์ตต่าง ๆ โดยอาจจะใช้ API ต่อไปนี้

- ส่งค่าเป็นไบท์ออกจากพอร์ต

  #include <asm/io.h>

  void outb(unsigned int value, unsigned short port)

  void outb_p(unsigned int value, unsigned short port)

- ส่งค่าเป็น word ออกจากพอร์ต

  #include <asm/io.h>

  void outw(unsigned int value, unsigned short port)

  void outw_p(unsigned int value, unsigned short port)

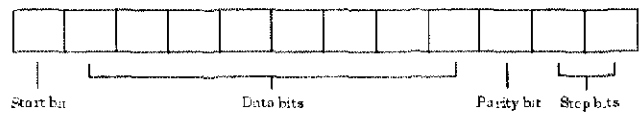
- รับค่าเป็นไบท์จากพอร์ต

  #include <asm/io.h>

  void inb(unsigned short port)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ติดต่อสื่อสารโดยผ่านพอร์ทอนุกรมและเป็นงานแบบเรียลไทม์อีกด้วย ซึ่งโปรโตคอลที่ทำการส่งนั้นอยู่ในรูปแบบดังรูปที่ 3.7



รูปที่ 3.1 แสดงโปรโตคอลเมื่อทำการส่งข้อมูลผ่านพอร์ตอนุกรม

3.4 ศึกษาการเขียนโปรแกรม

โครงงานนี้ประกอบไปด้วยโปรแกรมทั้งหมด 10 ไฟล์ด้วยกัน ซึ่งจะแบ่งออกเป็น 2 ประเภทใหญ่ ๆ ดังนี้

3.4.1 ประเภท Makefile

ในโครงงานนี้ได้มีไฟล์ประเภทนี้อยู่ด้วยกัน 3 ไฟล์ โดยที่ไฟล์ประเภทนี้เป็นการเขียนไฟล์ที่ช่วยในการอำนวยความสะดวกในการทำงานต่าง ๆ บนระบบปฏิบัติการลินุกซ์(ระบบเท็กซ์โหมด) ซึ่งไฟล์ประเภทนี้ช่วยในการทำคำสั่งต่าง ๆ หลาย ๆ คำสั่งด้วยการป้อนคำสั่งเพียงคำสั่งเดียวคือ "make" โดยเมื่อได้ป้อนคำสั่งแล้วระบบจะไปดูว่าในระบบ (ในไดเร็คตอรีเดียวกัน) มีไฟล์ Makefile อยู่หรือไม่ถ้ามีก็จะทำงานตามคำสั่งใน Makefile นั้นๆ ซึ่งชื่อไฟล์ "Makefile" สามารถเปลี่ยนเป็นชื่ออื่นได้เช่น "newMakefile" แต่ว่าเวลาเรียกคำสั่ง "make" จะต้องใช้คำสั่งดังนี้

"make –f newMakefile"

ในโครงงานนี้ได้มี Makefile อยู่ทั้งหมด 3 ไฟล์โดยโครงสร้างทั่วไปของ Makefile จะมีดั้งนี้

all: ชื่อโปรแกรมย่อยใน Makefile หรือ คำสั่งต่าง ๆ ที่ต้องการให้ทำงาน

---

void inb_p(unsigned short port)

- รับค่าเป็นไบท์ word ออกจากพอร์ต

#include <asm/io.h>

void inbw(unsigned int value, unsigned short port)

void *inbw_p(unsigned int value, unsigned short port)*

สำหรับฟังก์ชันที่มี "_p" ต่อท้ายจะเป็นฟังก์ชันที่ทำให้เกิดความล่าช้า (delay) น้อยๆ สำหรับ การเขียนและอ่านข้อมูลจากพอร์ต ซึ่งการล่าช้านี้อาจจะต้องการสำหรับ ISA ที่ช้า ๆ บนเครื่องที่รวดเร็วบางเครื่อง [12]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อโปรแกรมย่อย: ชื่อไฟล์ที่เกี่ยวข้องกับโปรแกรมย่อยนั้น ๆ
<---- กด Tab ---->คำสั่งที่ต้องการให้ทำ
ชื่อโปรแกรมย่อย: ชื่อไฟล์ที่เกี่ยวข้องกับโปรแกรมย่อยนั้น ๆ
<---- กด Tab ---->คำสั่งที่ต้องการให้ทำ
ชื่อโปรแกรมย่อย: ชื่อไฟล์ที่เกี่ยวข้องกับโปรแกรมย่อยนั้น ๆ
<---- กด Tab ---->คำสั่งที่ต้องการให้ทำ

เนื่องจากในโครงงานนี้ได้เขียนโปรแกรมที่เกี่ยวข้องกับระบบปฏิบัติการแบบเรียลไทม์ จึงทำให้ต้องนำเข้าไฟล์ "rtl.mk" ในทุก Makefile ซึ่งสิ่งที่อยู่ในไฟล์ "rtl.mk" นั้นจะเกี่ยวข้องกับสิ่งต่าง ๆ ที่จำเป็นต้องใช้ในการคอมไพล์ไฟล์ที่เกี่ยวข้องกับการทำงานแบบ เรียลไทม์โดยจะอยู่ในรูปของตัวแปรให้เรียกใช้ ซึ่งการประกาศจะใช้คำสั่ง "include" ดังนี้

```
"include /usr/rtlinux/rtl.mk"
```

นอกจากนั้นในทุก Makefile จะมีการใช้คำสั่งในการคอมไพล์ไฟล์ที่จำเป็น โดยจะมีการใช้คำสั่งในรูปแบบดังนี้

```
"$(CC) `rtl-config --สิ่งที่ต้องการ` -Wall -c ชื่อไฟล์.c –o ชื่อไฟล์.o "
```

${CC}	เป็นการเรียกใช้ตัวแปรซึ่งอยู่ภายใน Makefile เอง โดยในที่นี้จะมีค่าเป็น "gcc"
rtl-config	เป็นคำสั่งเพื่อที่จะเรียกเอาตัวเลือกต่าง ๆ ที่จำเป็นในการคอมไพล์โดยสิ่งต่างๆ จะอยู่ในไฟล์ rtl.mk
--สิ่งที่ต้องการ	เป็นชื่อตัวแปรที่อยู่ใน rtl.mk ซึ่งในที่นี้ใช้ 2 ตัวคือ "include" และ "cflags" โดย "include" จะใช้เมื่อต้องการคอมไพล์ไฟล์ที่เกี่ยวกับการสร้างงานที่มีการติดต่อกับงานที่มีการทำงานแบบเรียลไทม์(ใช้การติดต่อสื่อสารระหว่างกระบวนการ) และ "cflags" ใช้เมื่อต้องการคอมไพล์ไฟล์ที่สร้างงานที่เป็นแบบ เรียลไทม์
-Wall	เป็นคำสั่งใช้บอกถึงคำเตือนทั้งหมดที่เกิดขึ้น
-c	เป็นคำสั่งที่บอกว่าไฟล์ที่ต้องการจะคอมไพล์เป็นชื่อไฟล์ที่รับตามมา
-o	เป็นคำสั่งที่บอกว่าเมื่อคอมไพล์แล้วจะให้เก็บอยู่ในชื่อไฟล์ที่รับตามมา
<u>หมายเหตุ</u>	ในส่วนของ `rtl-config --สิ่งที่ต้องการ` ไม่จำเป็นจะต้องมีเมื่อเราคอมไพล์ไฟล์ที่ไม่เกี่ยวข้องกับการทำงานแบบ เรียลไทม์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 ไฟล์ที่เป็นการสร้างงานต่าง ๆ

ในโครงงานนี้ไฟล์ประเภทนี้แบ่งออกเป็นอีก 3 ประเภทคือ

3.4.2.1 ไฟล์ที่สร้างงานที่เป็นแบบเรียลไทม์

ไฟล์ประเภทนี้จะมีอยู่ด้วยกัน 3 ไฟล์คือ sent.c, receive.c และ rtfifo.c ซึ่งทั้ง 3 ไฟล์จะมีรูปแบบในการเขียนที่คล้ายคลึงกันโดนจะเขียนเป็นแบบเทรด (Thread) โดยที่การทำงานทั้ง 3 โปรแกรม เมื่อโมดูลเริ่มทำงานแต่ละโมดูลจะทำงานตามคำสั่งที่อยู่ในฟังก์ชัน *init_module()* โดยที่ในทั้ง 3 ไฟล์จะมีการทำงานคำสั่งดังนี้

```
rt_com_setup(0, 38400, RT_COM_PARITY_NONE, 1, 8);
return pthread_create (&thread, NULL, start_routine, 0);
rtf_create( DATA_FIFO, DATA_FIFO_SIZE );
```

- rt_com_setup(unsigned int *com*, unsigned int *baud*,
  unsigned int *parity*, unsigned int *stopbits*,
  unsigned int *wordlength*);

  เป็นคำสั่งที่ปรับค่าเริ่มต้นของการติดต่อสื่อสารโดยใช้โมดูลมาตรฐาน rt_com พบได้ใน sent.c บรรทัดที่ 28 และ 34 receive.c บรรทัดที่ 27 และ 33 rtfifo.c บรรทัดที่ 32 และ 39 จากตัวอย่างที่ยกมานี้ได้ทำการใส่ค่าเริ่มต้นให้เปิดพอร์ท COM1 จากการให้พารามิเตอร์ตัวแรกเป็น 0 ใช้อัตราความเร็วในการส่งเป็น 38400 ไบท์ต่อวินาที ไม่ใช้บิต Parity ใช้บิตหยุด(Stop bit) 2 บิต และ ส่งครั้งละ 8 บิท

- int pthread_create(pthread_t * thread, pthread_attr_t * attr,
  void *(*start_routine)(void *), void * arg);

  เป็นคำสั่งที่ให้สร้างเทรด(Thread) พบได้ใน sent.c บรรทัดที่ 29 receive.c บรรทัดที่ 28 rtfifo.c บรรทัดที่ 33 จากตัวอย่างที่ยกมาได้ทำการสร้างเทรดที่ชื่อ start_routine

- rtf_create(unsigned int *fifo*, int *size*);

  เป็นคำสั่งที่ทำการสร้างช่องทางในการติดต่อสื่อสารระหว่างกระบวนการ พบได้ใน rtfifo.c บรรทัดที่ 31 จากตัวอย่างที่ยกมาได้ทำการสร้างช่องทางใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การติดต่อสื่อสารระหว่างกระบวนการชื่อว่า DATA_FIFO โดยกำหนดขนาดของ FIFO คือ DATA_FIFO_SIZE

และเมื่อทำการสร้างเทรด(Thread) จะมีการปรับค่าเริ่มต้นของเทรดดังนี้

```
pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);
pthread_make_periodic_np (pthread_self(), gethrtime(), 100000000);
```

- pthread_setschedparam(pthread_t *thread*, int *policy*,
  const struct sched_param **param*);

  เป็นคำสั่งเพื่อให้มีการปรับค่าการจัดลำดับการทำงาน พบได้ใน sent.c บรรทัดที่ 12 receive.c บรรทัดที่ 11 rtfifo.c บรรทัดที่ 14 จากตัวอย่างที่ยกมาได้ทำการปรับค่าให้กับเทรดที่ถูกใช้งานอยู่ โดยเรียกคำสั่ง pthread_self() ซึ่งจะคืนค่าเป็นเทรดที่ทำงานอยู่ ณ ขณะนั้น และปรับวิธีการลำดับการทำงานเป็นวิธี FIFO และได้ทำการปรับค่าให้เป็นไปตามค่าที่ส่งไปในรูปแบบของ struct sched_param ซึ่งในที่นี้คือ p

- pthread_make_periodic_np(pthread_t *thread*, hrtime_t *start_time*,
  hrtime_t *period*);

  เป็นคำสั่งที่สั่งให้เทรดเริ่มทำงานเรียลไทม์แบบรายคาบ พบได้ใน sent.c บรรทัดที่ 13 receive.c บรรทัดที่ 12 rtfifo.c บรรทัดที่ 15 จากตัวอย่างที่ยกมาได้ทำการสั่งให้เทรดทำงานโดยเรียกคำสั่ง pthread_self() ซึ่งจะคืนค่าเป็นเทรดที่ทำงานอยู่ โดยเริ่มต้นที่เวลาในขณะนั้น ซึ่งจะได้จากการเรียกคำสั่ง gethrtime() ซึ่งจะคืนค่ามาเป็นเวลาของระบบที่ และได้ให้คาบเวลาเท่ากับ 0.1 วินาที(10^8 นาโนวินาที = 0.1 วินาที)

หลังจากนั้นทุก ๆ ไฟล์ได้เริ่มเข้าสู่การทำงานแบบรายคาบและถูกพักงาน ซึ่งเทรดจะถูกพักจนกว่าจะถึงเวลาที่ครบคาบเวลาและจะทำตามคำสั่งหลังจากนั้นต่อไปโดยที่ในส่วนของไฟล์ sent.c จะทำการส่งค่าผ่านพอร์ตอนุกรมแต่ถ้าเป็น receive.c และ rtfifo.c จะรับค่าจากพอร์ตอนุกรม นอกจากนั้นในไฟล์ rtfifo.c ยังมีการนำค่าเวลา ณ ขณะนั้นเก็บไว้ในตัวแปรเพื่อที่จะส่งค่าเข้าไปใน FIFO ที่เปิดไว้สำหรับการติดต่อสื่อสารระหว่างกระบวนการ และท้ายที่สุดทุกไฟล์จะพิมพ์ค่าลงสู่เคอร์เนล โดยที่คำสั่งที่ถูกเรียกใช้มีดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pthread_wait_np();
rt_com_write( 0, &tmp , 1 );
rt _com_read(0, buf, sizeof(buf));
time(NULL);
rtf_put( DATA_FIFO, &data, sizeof(data) );
rtl_printf("round : >> %d << and data is >> %s\n",i,buf);
```

- pthread_wait_np(void);

  เป็นคำสั่งที่ทำให้งานใด ๆ (เทรด)ถูกพักงานจนกว่าจะครบคาบที่กำหนด พบได้ใน sent.c บรรทัดที่ 17 receive.c บรรทัดที่ 16 rtfifo.c บรรทัดที่ 18

- rt_com_write(unsigned int *com*, char **ptr*, int *cnt*);

  เป็นคำสั่งที่ส่งค่าออกจากพอร์ตอนุกรมซึ่งฟังก์ชันนี้อยู่ในโมดูลมาตรฐาน rt_com ซึ่ง พบได้ใน sent.c บรรทัดที่ 21 จากตัวอย่างที่ยกมา เป็นการส่งค่าออกจากพอร์ต COM1 (กำหนดค่าเป็น 0) ค่าที่ส่งไปคือค่าที่ถูกชี้ด้วยแอดเดรสของ tmp ส่วนพารามิเตอร์ตัวสุดท้ายก็เป็นจำนวนไบท์ที่ส่งต่อครั้งคือ 1 ไบท์ต่อวินาที

- rt_com_read(unsigned int *com*, char **ptr*, int *cnt*);

  เป็นคำสั่งที่ไว้อ่านรับจากพอร์ตอนุกรมซึ่งฟังก์ชันนี้อยู่ในโมดูลมาตรฐาน rt_com พบได้ใน receive.c บรรทัดที่ 17 rtfifo.c บรรทัดที่ 19 จากตัวอย่างที่ยกมา พารามิเตอร์ตัวแรกจะเป็นพอร์ตในที่นี้ได้กำหนดให้รับค่าจากพอร์ท COM1(กำหนดค่าเป็น 0) ซึ่งค่าที่รับนั้นจะถูกเก็บไว้ในตัวแปร buf ส่วนพารามิเตอร์ตัวสุดท้ายก็เป็นจำนวนไบท์ที่รับต่อครั้งซึ่งคือ 1 ไบท์ต่อวินาที

- time(time_t **tloc*);

  เป็นคำสั่งที่เมื่อเรียกแล้วจะคืนค่ามาเป็นเวลาของระบบซึ่งมีหน่วยเป็นวินาทีนับจากวันที่ 1 มกราคม ปี ค.ศ. 1970พบได้ใน batch.c บรรทัดที่ 18 rtfifo.c บรรทัดที่ 20 cube.c บรรทัดที่ 11 และ 23 Matrix.c บรรทัดที่ 31 และ 42

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- rtf_put(unsigned int *fifo*, char * *buf*, int *count*);

  เป็นคำสั่งเพื่อเขียนค่าลงในช่องทางในการติดต่อสื่อสารระหว่างกระบวนการ พบได้ใน rtfifo.c บรรทัดที่ 23 จากตัวอย่างที่ยกมาได้ทำการใส่ค่าที่ถูกเก็บไว้ใน data ขนาดเท่ากับขนาดของ date ซึ่งได้จาก sizeof(data) โดยส่งไปยังช่องทางการติดต่อสื่อสารระหว่างกระบวนการที่ชื่อว่า DATA_FIFO

- rtl_printf(const char **format*, ...);

  เป็นคำสั่งเพื่อให้แสดงข้อความในเคอร์เนล พบได้ใน sent.c บรรทัดที่ 22 receive.c บรรทัดที่ 21 rtfifo.c บรรทัดที่ 24

จุดสิ้นสุดของการทำงานของแต่ละโมดูลจะสิ้นสุดเมื่อมีการป้อนคำสั่ง "rmmod" โดยการทำงานจะทำตามคำสั่งในฟังก์ชัน cleanup_module() ซึ่งในทุกไฟล์จะเริ่มจาก ทำลายเทรดที่มีอยู่อีกทั้งและถ้าเป็นไฟล์ rtfifo.c จะทำการทำลายช่องทางในการติดต่อสื่อสารระหว่างกระบวนการและสุดท้ายจะพิมพ์ค่าออกทางเคอร์เนล โดยที่คำสั่งที่ใช้มีดังนี้

```
pthread_delete_np (thread);
rtf_destroy( DATA_FIFO );
printk("Bye\n");
```

- pthread_delete_np(pthread_t *thread*);

  เป็นคำสั่งที่ไว้ทำลายเทรดที่มีอยู่โดยพบได้ใน sent.c บรรทัดที่ 32 receive.c บรรทัดที่ 31 rtfifo.c บรรทัดที่ 38 จากตัวอย่างที่ยกมานี้ได้ทำลายเทรดที่ชื่อว่า thread

- rtf_ destroy(unsigned int *fifo*);

  เป็นคำสั่งที่เอาไว้ทำลายช่องทางในการติดต่อสื่อสารระหว่างกระบวนการ พบได้ใน rtfifo.c บรรทัดที่ 37 จากตัวอย่างที่ยกมาได้ทำการทำลายช่องทางในการติดต่อสื่อสารที่ชื่อว่า DATA_FIFO

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- printk(const char *fmt,);

  เป็นคำสั่งเพื่อให้แสดงข้อความในเคอร์เนล พบได้ใน sent.c บรรทัดที่ 33 receive.c บรรทัดที่ 32 rtfifo.c บรรทัดที่ 40

3.4.2.2 ไฟล์ที่สร้างงานที่เกี่ยวข้องกับงานแบบเรียลไทม์

ในที่นี้มี 2 ไฟล์ที่ทำงานเกี่ยวกับการติดต่อสื่อสารระหว่างกระบวนการก็คือไฟล์ batch.c และ common.h ซึ่งไฟล์ batch.c จะทำการนำค่าที่ถูกใส่ใน FIFO โดยไฟล์ rtfifo.c ออกมาจาก FIFO ซึ่งในตอนเริ่มต้นจะทำการเปิดช่องทางที่ใช้รับข้อมูล(FIFO) ถ้าเปิดช่องทางการติดต่อนี้ไม่สำเร็จ และจะทำให้ออกจากโปรแกรมนี้ไป ในทางกลับกันถ้าเปิดสำเร็จก็จะเข้าสู่การอ่านค่าจาก FIFO ซึ่งถ้ามีค่าใน FIFO ก็จะถูกอ่านออกมาแต่ถ้าไม่มีค่าใด ๆ ใน FIFO โปรแกรมก็จะรอจนกว่าจะมีค่าเข้ามาใน FIFO แล้วนำค่านั้นออกต่อไป โดยที่ค่าที่ได้ออกมานั้นจะเป็นค่าเวลาเมื่อ rtfifo.c ทำการใส่ค่าลงใน FIFO ซึ่งโปรแกรมจะนำที่ได้นี้แสดงออกหน้าจอ เมื่อทำการแสดงค่าหรือนำค่าออกมาจนกระทั่งได้ตามค่าที่กำหนดไว้(ในที่นี้กำหนดให้ทำการรับค่า 500,000 ครั้ง)จะทำการปิดช่องทางในการติดต่อสื่อสารระหว่างกระบวนการเป็นอันจบโปรแกรม batch.c ซึ่งในโปรแกรมนี้จะมีการทำคำสั่งดังนี้คือ

```
open("/dev/rtf1", O_RDONLY))
read(fd_data, &data, sizeof(data));
close( fd_data );
```

- open(const char *file, int mode /*, int permissions */);

  เป็นคำสั่งที่มีไว้เปิดช่องทางในการติดต่อสื่อสารระหว่างกระบวนการ โดยที่ค่าที่ได้จากฟังก์ชันนี้ถ้าสามารถเปิดช่องทางการติดต่อนี้สำเร็จจะคืนค่าเป็นค่าที่มากกว่าหรือเท่ากับ 0 แต่ถ้าเปิดช่องทางไม่สำเร็จจะคืนค่ามาเป็น 0 พบได้ใน batch.c บรรทัดที่ 12 จากตัวอย่างที่ยกมา เราได้ทำการเปิดช่องทางในการคิดต่อสื่อสาร โดยใช้ประเภทที่ใช้คืออ่านเพียงอย่างเดียว

- read(int fd, void *buffer, size_t length);

  เป็นคำสั่งที่มีไว้เพื่อที่จะอ่านข้อมูลจากช่องทางในการติดต่อสื่อสารระหว่างกระบวนการ ซึ่งคำสั่งนี้ถ้าในขณะนั้นไม่มีค่าที่อยู่ใน FIFO จะทำให้เกิดการคอยจนกว่าจะมีข้อมูลใน FIFO เพื่อที่จะอ่านข้อมูลออกจาก FIFO

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถ พบได้ใน batch.c บรรทัดที่ 17 จากตัวอย่างที่ยกมา เราได้ทำอ่านข้อมูลจากช่องทางในการคิดต่อสื่อสารชื่อว่า fd_data โดยเมื่ออ่านได้แล้วจะทำการเก็บไว้ใน data ซึ่งจะอ่านมาครั้งละเท่ากับขนาดของ data ซึ่งได้จาก *sizeof(data)*

- close(int fd);

  เป็นคำสั่งที่มีไว้เพื่อทำการปิดช่องทางการติดต่อสื่อสารระหว่างกระบวนการเมื่อเลิกใช้งานแล้ว พบได้ใน batch.c บรรทัดที่ 22 จากตัวอย่างที่ยกมาได้ทำการปิด

ในส่วนของ common.h จะเป็นไฟล์ที่เก็บเอาโครงสร้างที่ใช้ในการส่งค่าผ่าน FIFO อีกทั้งยังเป็นส่วนประกาศค่าที่มีไว้เพื่อใช้ในไฟล์ rtfifo.c เพื่อใช้ในการปรับค่าเริ่มต้นของการสร้างช่องทางในการติดต่อสื่อสาร อีกด้วย

3.4.2.3 ไฟล์ที่สร้างงานเพื่อวัดผลเวลา

ไฟล์ประเภทนี้จะมีทั้งหมด 2 ไฟล์ โดยที่จะเป็นไฟล์ที่มีความเกี่ยวข้องกันซึ่งก็คือการทำงานของการวนรอบ 3 ชั้นและการคูณ Matrix ซึ่งหลักการทำงานนั้นจะมีการวนรอบทั้งหมด 3 ชั้นซึ่งแต่ละชั้นจะมีรอบที่เท่ากัน เช่น

```
for (i=0;i<1000;i++)
{       for (j=0;j<1000;j++)
        {       for (k=0;k<1000;k++)
                {
                temp++;
                a = i*j*k;
                }
        }
}
```

ตัวอย่างโปรแกรมนี้อยู่ในไฟล์ cube.c ซึ่งจะมีการวนรอบ 3 ชั้นซึ่งจะทำงานชั้นละ 1,000 ครั้ง ดังนั้นจำนวนครั้งทั้งหมดจึงเท่ากับ 1,000,000,000 ครั้ง ซึ่งในโปรแกรมการคูณ Matrix ก็เช่นกันแต่ว่าในการวนรอบชั้นในสุดจะมีการคำนวณเกิดขึ้นโดย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Ans[i][j] = Ans[i][j] + ((A[i][k])*(B[k][j]));
```

จากการที่โปรแกรม cube.c นั้นเป็นการทำงานที่เรียกได้ว่าใช้การทำงานของหน่วยประมวลผลกลางเพียงอย่างเดียว(CPU Intensive) ถ้าเทียบกับโปรแกรม matrix.c จะเป็นแบบที่เราถือว่าเป็นแบบ generic programming ซึ่งหมายถึงการเขียนโปรแกรมทั่ว ๆ ไปที่มีการทำงานของการสลับหน้าของหน่วยความจำหลัก(Memory) อีกทั้งยังมีการทำงานทั่ว ๆ ไปของหน่วยประมวลผลกลาง ซึ่งเมื่อสังเกตจากความซับซ้อนของข้อมูลที่ใช้ใน matrix.c (การคูณ Matrix ใช้อาร์เรย์ขนาด 1,000 คูณ 1,000 จำนวน 2 อาร์เรย์ และผลลัพธ์สำหรับการคูณ Matrix อีก 1 อาร์เรย์) จะทำให้เกิดภาระของการทำงานมากขึ้น ทางผู้จัดทำจึงได้ทำการเก็บค่าเวลาที่ใช้โดยเริ่มเก็บค่าเวลาตั้งแต่ก่อนที่จะเข้าทำการวนรอบจนกระทั่งทำการวนรอบจบ และจะทำการพิมพ์ค่าออกจากหน้าจอของโปรแกรม cube.c และ matrix.c เพื่อที่จะนำผลเวลานั้นมาวิเคราะห์และสรุปผลต่อไป

3.5 วิธีการรันและยกเลิกการทำงานของโปรแกรม

3.5.1 การรันโปรแกรมและโมดูล

หลังจากทำการคอมไพล์ไฟล์ที่เป็นนามสกุล ".c" ซึ่งจะได้แบบที่เป็นนามสกุล ".o" (เฉพาะงานที่เป็นแบบเรียลไทม์[3]) และใช้คำสั่ง "insmod" โดยคำสั่งนี้จะเป็นการเพิ่มโมดูล (อ็อบเจกต์ที่ได้จากการคอมไพล์) เข้าไปในเคอร์เนล[4]ตัวอย่าง เช่นเมื่อทำการคอมไพล์ sent.c จะได้ sent.o ซึ่งในที่นี้เรียกว่าโมดูล และทำการป้อนคำสั่งดังนี้

```
"insmod sent.o"
```

ในส่วนของ Cube.c, Matrix.c และ Batch.c เมื่อทำการคอมไพล์เป็นนามสกุล ".o" แล้วเมื่อต้องการที่จะเรียกให้โปรแกรมเหล่านี้ทำงานทำได้โดยเรียกคำสั่ง "./"แล้วตามด้วยชื่อโปรแกรม เช่น เมื่อทำการคอมไพล์โปรแกรม Cube.c จะได้เป็น Cube.o เมื่อจะทำการรันโปรแกรมจะเรียกคำสั่งดังนี้

```
"./Cube.o"
```

[3] งานที่เป็นเรียลไทม์เรียกว่า kernel module

[4] User program จะทำงานอยู่ใน user space แต่ kernel module ทำงานอยู่ใน kernel space

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5.2 การหยุดการทำงานของโปรแกรมและโมดูล

การหยุดการทำงานของโมดูลนั้นจะเป็นการนำโมดูลนั้น ๆ ออกจากเคอร์เนล โดยการเรียกคำสั่ง "rmmod" แล้วตามด้วยชื่อของโมดูลที่ต้องการ เช่นเมื่อต้องการหยุดโมดูลชื่อ "sent" ซึ่งในขณะนั้นกำลังทำงานอยู่ในเคอร์เนล จะทำการป้อนคำสั่งดังนี้

"rmmod sent"

ในส่วนของ Cube.c, Matrix.c และ Batch.c ในขณะที่โปรแกรมเหล่านี้ทำการรันอยู่ แล้วต้องการที่จะหยุดการทำงานเหล่านี้โดยกดปุ่ม Ctrl และ ปุ่ม C พร้อมกัน

3.6 การเรียกดูข้อมูลในเคอร์เนล

ในการเรียกคำสั่ง printk() และ rtl_printf() จะเป็นการส่งข้อความเข้าไปยังเคอร์เนลบัฟเฟอร์ซึ่งจะยังไม่แสดงผลทันทีถ้าทำงานในแบบกราฟฟิกโหมดของลินุกซ์(ในที่นี้ใช้กราฟฟิกโหมด) แต่ถ้าทำงานในเท็กซ์โหมดจะแสดงผลทันที เมื่อเราต้องการที่จะดูข้อความในเคอร์เนลบัฟเฟอร์จะใช้คำสั่งดังนี้

"dmesg"

3.7 ขั้นตอนในการรันโปรแกรมที่เป็นแบบเรียลไทม์

เนื่องจากในปัญหาพิเศษนี้ได้มีการส่งและรับผ่านพอร์ตอนุกรมซึ่งได้ใช้โมดูลมาตรฐาน rt_com อีกทั้งเคอร์เนลยังเป็นอาร์ทีลินุกซ์ซึ่งทั้ง 2 อย่างข้างต้นจะมีวิธีการเรียกใช้ที่เฉพาะเจาะจง ซึ่งสามารถอธิบายการเรียกใช้และยกเลิกการทำงานได้เป็นขั้นตอนดังนี้

1) เรียกคำสั่ง "rtlinux start" เพื่อที่จะเรียกโมดูลต่าง ๆ ของอาร์ทีลินุกซ์ให้ทำงาน

2) เข้าไปที่ไดเร็คตอรีของ rt_com แล้วทำการเรียกคำสั่ง "make test" และ "insmod rt_com" เพื่อที่จะเรียกคำสั่งต่าง ๆ ในโมดูลมาตรฐาน rt_com

3) แทรกโมดูลต่าง ๆ ที่ต้องการจะให้ทำงานเข้าไปในเคอร์เนล

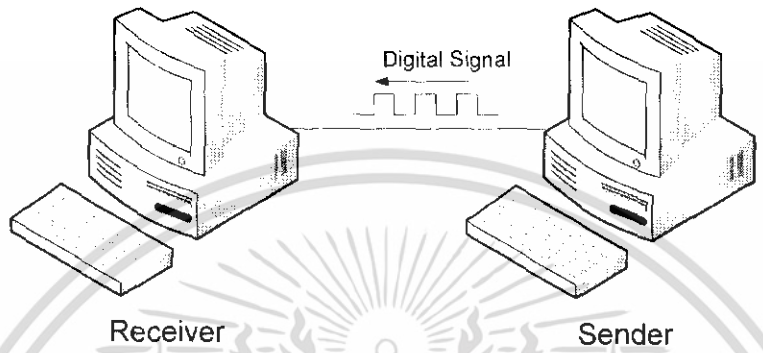
4) เมื่อต้องการจะจบการทำงานให้เริ่มจากการนำโมดูลที่ทำงานอยู่และใช้คำสั่งในโมดูล rt_com ออกจากเคอร์เนลก่อน จากนั้นจึงนำโมดูล rt_com ออกจากเคอร์เนล

5) ทำการนำโมดูลของอาร์ทีลินุกซ์ออกจากเคอร์เนล โดยเรียกคำสั่ง "rtlinux stop"

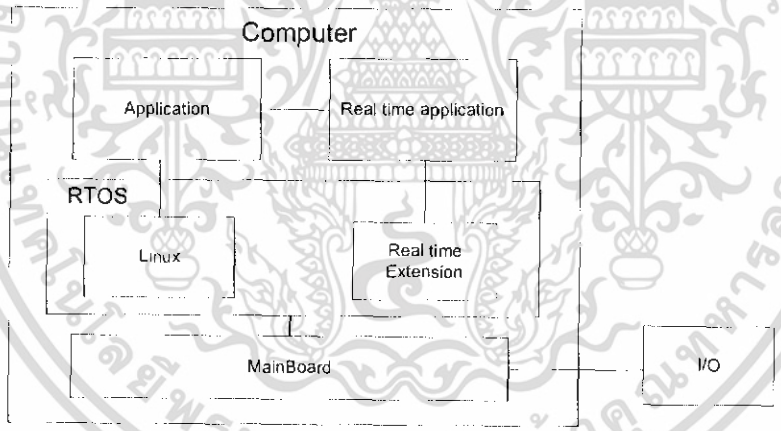
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8 กำหนดขอบเขตที่จะทำสร้างระบบส่งผ่านข้อมูลที่เป็นแบบเรียลไทม์

ในการกำหนดขอบเขต จะมุ่งไปที่ผลเวลาของงานที่ทำงานพร้อมกับงานที่เป็นแบบเรียลไทม์ที่ทำการรับข้อมูลจากพอร์ตอนุกรมซึ่งโปรแกรมทั้งหมดนี้ถูกสร้างขึ้นมาด้วยภาษาซีเพื่อที่ทำให้การศึกษาระบบเป็นไปได้ง่ายขึ้น จะสนใจแค่ในส่วนของการรับข้อมูลและทำการแสดงผลทางหน้าจอเพียงอย่างเดียว



รูปที่ 3.2 ระบบคอมพิวเตอร์แบบเรียลไทม์ในภาพรวม

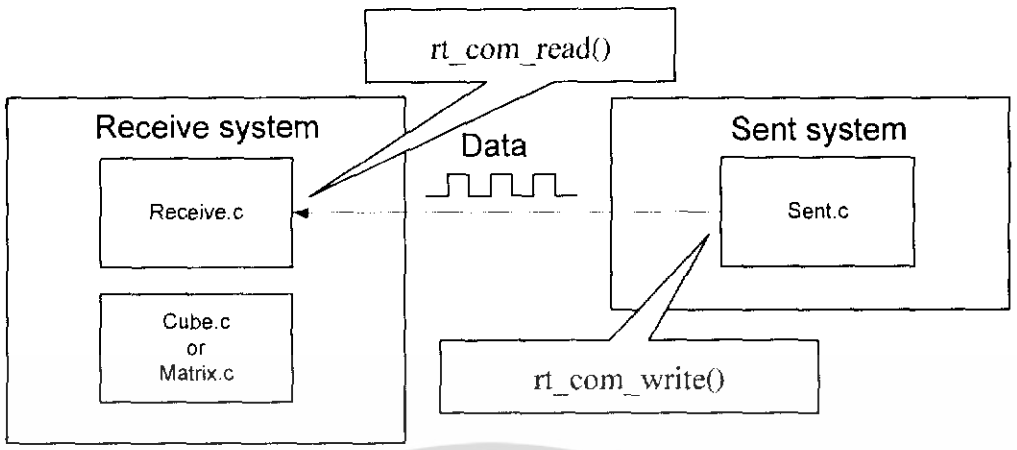


รูปที่ 3.3 โครงสร้างระบบคอมพิวเตอร์แบบเรียลไทม์ในรายละเอียดภายใน

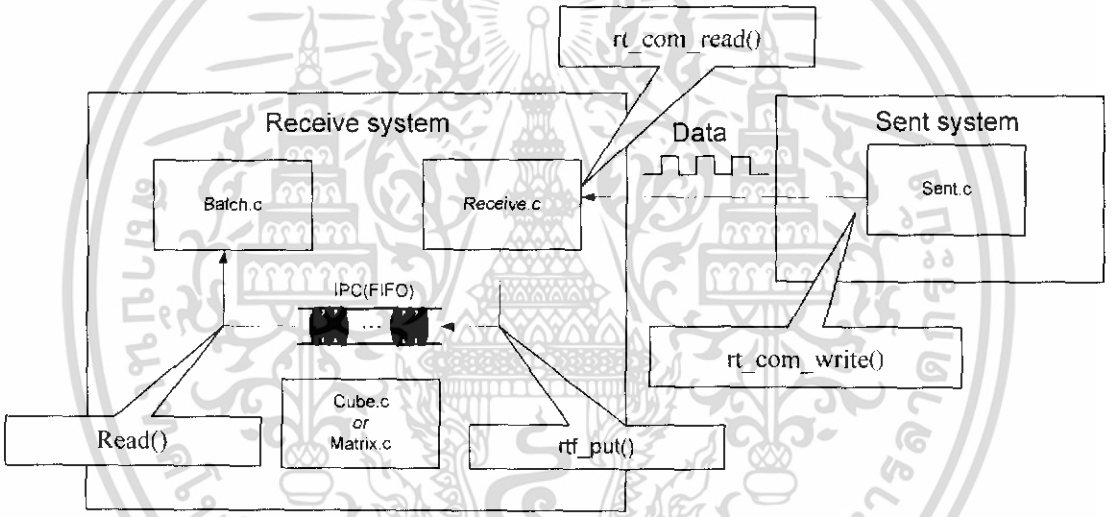
จากรูปที่ 3.2 แสดงถึงการส่งผ่านสัญญาณจากเครื่องคอมพิวเตอร์เครื่องหนึ่งไปยังเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งซึ่งมองในภาพรวม และจากรูปที่ 3.3 ในส่วนที่เป็นงานเรียลไทม์ คือส่วนที่ได้ทำการเขียนโปรแกรมเพื่อทำการส่งข้อมูลจากคอมคอมพิวเตอร์เครื่องหนึ่งไปยังเครื่องคอมพิวเตอร์อีกเครื่องหนึ่งแล้วทำการแสดงผลผ่านทางหน้าจอคอมพิวเตอร์ซึ่งจะถูกจัดการโดยระบบปฏิบัติการแบบเรียลไทม์ และ เมนบอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.9 วิธีการทดลอง



รูปที่ 3.4 รูปแสดงระบบการทำงานเมื่อไม่มีการติดต่อสื่อสารของกระบวนการ



รูปที่ 3.5 รูปแสดงระบบการทำงานเมื่อมีการติดต่อสื่อสารของกระบวนการ

เนื่องจากความต้องการที่จะศึกษาเกี่ยวกับการทำงานของระบบปฏิบัติการที่เป็นแบบ เรียลไทม์ โดยที่เราทำการสร้างงานที่เป็น เรียลไทม์ ให้รับค่าจากพอร์ตอนุกรม ซึ่งได้แบ่งการทดลองออกเป็น 3 ขั้นตอนคือ

1. การศึกษาทางทฤษฎีเกี่ยวกับระบบที่ทำงานแบบ เรียลไทม์ และศึกษาการติดตั้งระบบที่ทำงานแบบเรียลไทม์ ซึ่งก็คืออาร์ทีลินุกซ์นั่นเอง

2. ติดตั้งลินุกซ์และติดตั้งส่วนขยายเพื่อให้ลินุกซ์สามารถทำงานแบบเรียลไทม์ อย่างแข็งได้ (Hard Real Time)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ทำการทดลองโดยสร้างงานที่ให้รับค่าจากพอร์ตอนุกรมซึ่งทำงานพร้อมกับโปรแกรมคำนวณทางคณิตศาสตร์โดยแบ่งออกเป็น 2 กรณีคือ

3.1 ทดสอบเวลาที่ทำงานเสร็จด้วยโปรแกรมที่ทำงานด้วยการวนรอบ 3 ชั้นด้วยรอบที่เท่ากัน(Cube program)

3.2 ทดสอบเวลาที่ทำงานเสร็จด้วยโปรแกรมการคูณ Matrix (Multiplication Matrix program)

โปรแกรมคำนวณทางคณิตศาสตร์มีไว้เพื่อที่จะทำให้เกิดการแย่งหน่วยประมวลผลกลางทำงานหรือเกิดโหลด(Load) และในขั้นตอนที่ 3 นี้เองในแต่ละกรณีได้ถูกแบ่งออกเป็นย่อย ๆ อีกอย่างละ 5 กรณีคือ

1) ให้โปรแกรมทางคณิตศาสตร์ทำงานเพียงอย่างเดียว

2) ให้โปรแกรมทางคณิตศาสตร์ทำงานพร้อมกับให้มีการทำงานของงานที่เป็นแบบเรียลไทม์ โดยใช้การจัดลำดับในการทำงานแบบมาก่อนได้ก่อน (First-Come, First-Served Scheduling)

3) ให้โปรแกรมทางคณิตศาสตร์ทำงานพร้อมกับให้มีการทำงานของงานที่เป็นแบบเรียลไทม์โดยใช้การจัดลำดับในการทำงานแบบเวียนเทียน (Round-Robin Scheduling)

4) ให้โปรแกรมคำนวณทางคณิตศาสตร์ที่เหมือนกันทำงานพร้อมกัน 2 โปรแกรมพร้อมกัน โดยไม่มีการทำงานของงานที่เป็นแบบเรียลไทม์

5) ให้โปรแกรมทางคณิตศาสตร์ทำงานพร้อมกับโปรแกรมที่ไม่ใช่โปรแกรมแบบเรียลไทม์ติดต่อสื่อสารกับโปรแกรมที่เป็นโปรแกรมแบบ เรียลไทม์โดยผ่านการสื่อสารระหว่างกระบวนการ (Interprocess Communication, IPC) ที่อาร์ทีลินุกซ์ ได้จัดให้คือผ่านทาง FIFO โดยได้ทดสอบใน 2 กรณีคือ

5.1 ใช้การจัดลำดับการทำงานแบบเวียนเทียน (Round-Robin Scheduling)

5.2 ใช้การจัดลำดับการทำงานมาก่อนได้ก่อน (First-Come, First-Served Scheduling)

หลังจากได้ผลการทดลองแล้วจึงนำมาสรุปผลและวิเคราะห์ผลว่างานที่เป็นแบบเรียลไทม์มีผลกระทบต่องานที่ไม่เป็นแบบเรียลไทม์หรือไม่อย่างไร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
ผลการทดลองและอภิปรายผล

เนื่องจากความต้องการที่จะศึกษาเกี่ยวกับการทำงานของระบบปฏิบัติการที่เป็นแบบเรียลไทม์ (Real Time) โดยที่เราทำการสร้างงานที่เป็นแบบเรียลไทม์ให้รับค่าจากพอร์ตอนุกรมซึ่งได้แบ่งการทดลองออกเป็น 3 ขั้นตอนคือ

1. การศึกษาทางทฤษฎีเกี่ยวกับระบบที่ทำงานแบบเรียลไทม์และศึกษาการติดตั้งระบบที่ทำงานแบบเรียลไทม์ซึ่งก็คืออาร์ทีลินุกซ์นั่นเอง

2. ติดตั้งลินุกซ์และติดตั้งส่วนขยายเพื่อให้ลินุกซ์สามารถทำงานแบบเรียลไทม์อย่างแข็งได้ (Hard Real Time)

3. ทำการทดลองโดยสร้างงานที่ให้รับค่าจากพอร์ตอนุกรมซึ่งทำงานพร้อมกับโปรแกรมคำนวณทางคณิตศาสตร์ โดยแบ่งออกเป็น 2 กรณีคือ

3.1 ทดสอบเวลาที่ทำงานเสร็จด้วยโปรแกรมที่ทำงานด้วยการวนรอบ 3 ชั้นด้วยรอบที่เท่ากัน(Cube program)

3.2 ทดสอบเวลาที่ทำงานเสร็จด้วยโปรแกรมการคูณ Matrix (Multiplication Matrix program)

และในขั้นตอนที่ 3 นี้เองในแต่ละกรณีได้ถูกแบ่งออกเป็นย่อย ๆ อีกอย่างละ 5 กรณีคือ

1) ให้โปรแกรมทางคณิตศาสตร์ทำงานเพียงอย่างเดียว

2) ให้โปรแกรมทางคณิตสาสตร์ทำงานพร้อมกับให้มีการทำงานของงานที่เป็นแบบเรียลไทม์(Real Time Task) โดยใช้การจัดลำดับในการทำงานแบบมาก่อนได้ก่อน (First-Come, First-Served Scheduling)

3) ให้โปรแกรมทางคณิตสาสตร์ทำงานพร้อมกับให้มีการทำงานของงานที่เป็นแบบเรียลไทม์(Real Time Task) โดยใช้การจัดลำดับในการทำงานแบบเวียนเทียน (Round-Robin Scheduling)

4) ให้โปรแกรมคำนวณทางคณิตศาสตร์ที่เหมือนกันทำงานพร้อมกัน 2 โปรแกรมพร้อมกัน โดยไม่มีการทำงานของงานที่เป็นแบบเรียลไทม์

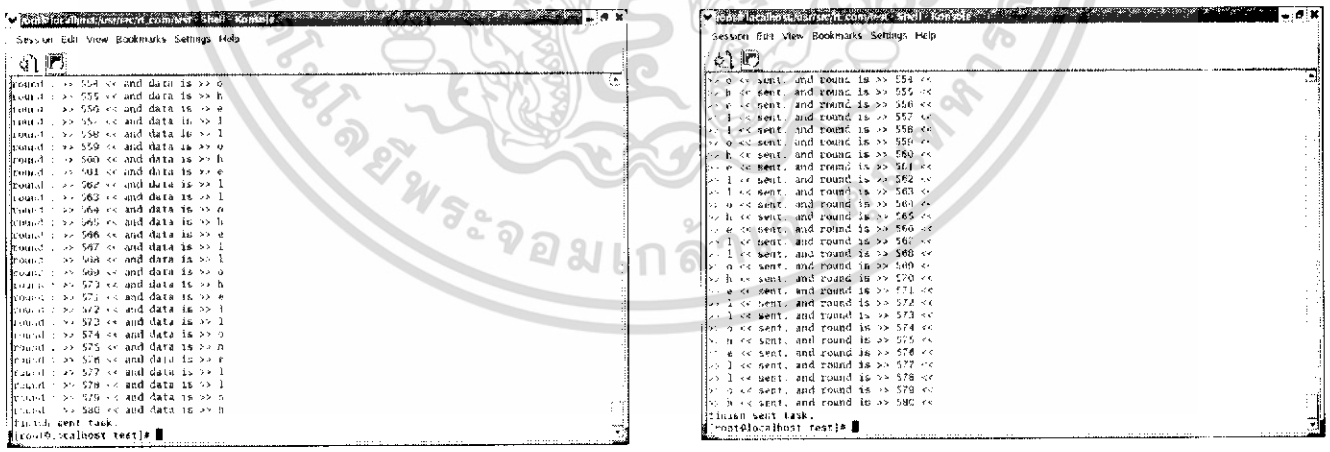
5) ให้โปรแกรมทางคณิตศาสตร์ทำงานพร้อมกับโปรแกรมที่ไม่ใช่โปรแกรมแบบที่ทำงานแบบเรียลไทม์ติดต่อสื่อสารกับโปรแกรมที่เป็นโปรแกรมแบบเรียลไทม์โดยผ่านการสื่อสารระหว่างกระบวนการ(Interprocess Communication, IPC) ที่ทางอาร์ทีลินุกซ์ได้จัดให้คือผ่านทาง FIFO โดยได้ทดสอบใน 2 กรณีคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ใช้การจัดลำดับการทำงานแบบเวียนเทียน (Round-Robin Scheduling)
- ใช้การจัดลำดับการทำงานมาก่อนได้ก่อน (First-Come, First-Served Scheduling)

```
round = >> 3 << and time >> 12.000000 <<
round = >> 4 << and time >> 12.000000 <<
round = >> 5 << and time >> 12.000000 <<
round = >> 6 << and time >> 12.000000 <<
round = >> 7 << and time >> 12.000000 <<
round = >> 8 << and time >> 12.000000 <<
round = >> 9 << and time >> 12.000000 <<
round = >> 10 << and time >> 12.000000 <<
round = >> 11 << and time >> 12.000000 <<
round = >> 12 << and time >> 12.000000 <<
round = >> 13 << and time >> 12.000000 <<
round = >> 14 << and time >> 12.000000 <<
round = >> 15 << and time >> 12.000000 <<
round = >> 16 << and time >> 12.000000 <<
round = >> 17 << and time >> 12.000000 <<
round = >> 18 << and time >> 12.000000 <<
round = >> 19 << and time >> 12.000000 <<
round = >> 20 << and time >> 12.000000 <<
[root@localhost src]# ./cube.o
round = >> 1 << and time >> 12.000000 <<
round = >> 2 << and time >> 12.000000 <<
round = >> 3 << and time >> 12.000000 <<
round = >> 4 << and time >> 12.000000 <<
round = >> 5 << and time >> 12.000000 <<
round = >> 6 << and time >> 12.000000 <<
round = >> 7 << and time >> 12.000000 <<
round = >> 8 << and time >> 12.000000 <<
round = >> 9 << and time >> 12.000000 <<
round = >> 10 << and time >> 12.000000 <<
round = >> 11 << and time >> 12.000000 <<
round = >> 12 << and time >> 12.000000 <<
round = >> 13 << and time >> 12.000000 <<
round = >> 14 << and time >> 12.000000 <<
round = >> 15 << and time >> 12.000000 <<
round = >> 16 << and time >> 12.000000 <<
round = >> 17 << and time >> 12.000000 <<
round = >> 18 << and time >> 12.000000 <<
round = >> 19 << and time >> 12.000000 <<
round = >> 20 << and time >> 12.000000 <<
[root@localhost src]#
```

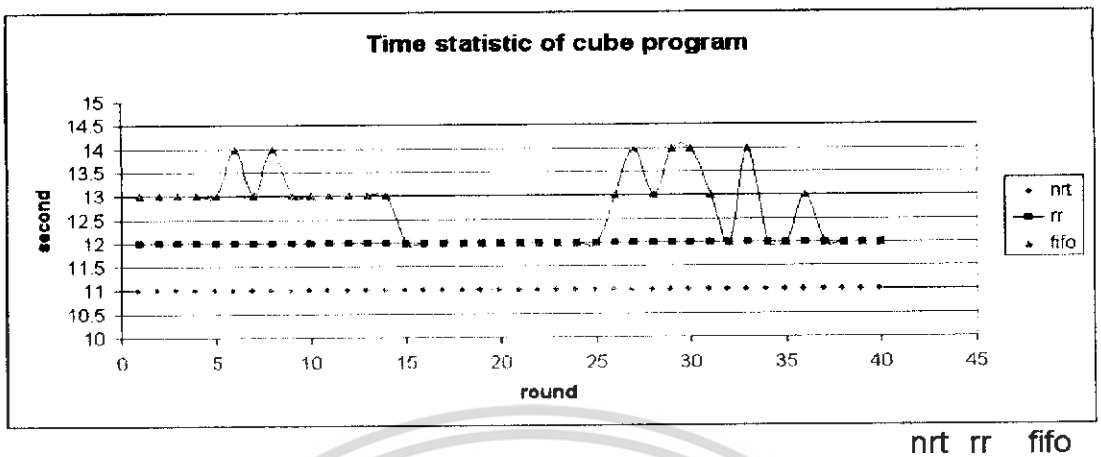
รูปที่ 4.1 หน้าจอผลการทดลองโปรแกรมการคำนวณทางคณิตศาสตร์



รูปที่ 4.2 หน้าจอผลการทดลองโปรแกรมรับ(ซ้าย)-ส่ง(ขวา)ข้อมูลผ่านทางพอร์ตอนุกรม

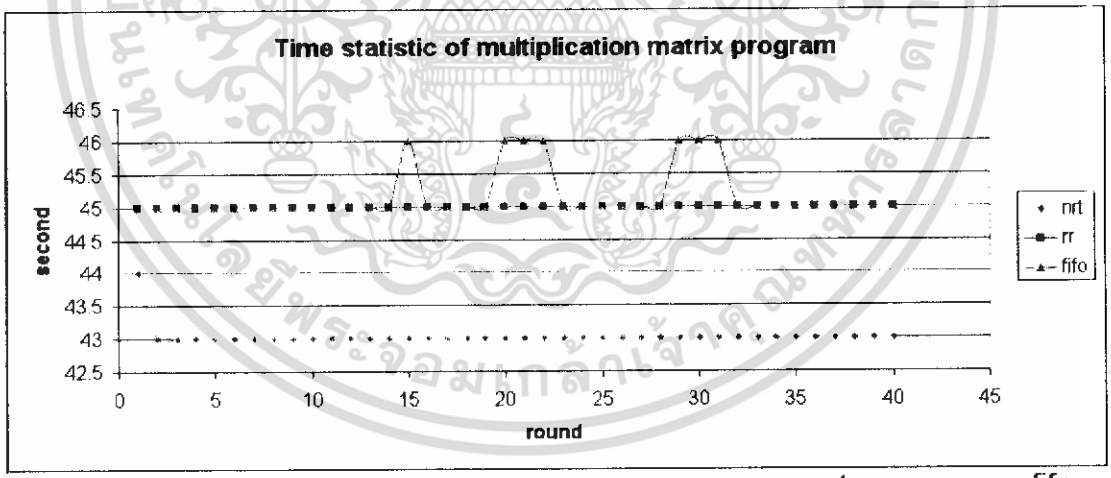
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1 ผลการทดลอง



	nrt	rr	fifo
average	11	12	12.7
sd	0	0	0.72
mode	11	12	12
mean	11	12	13

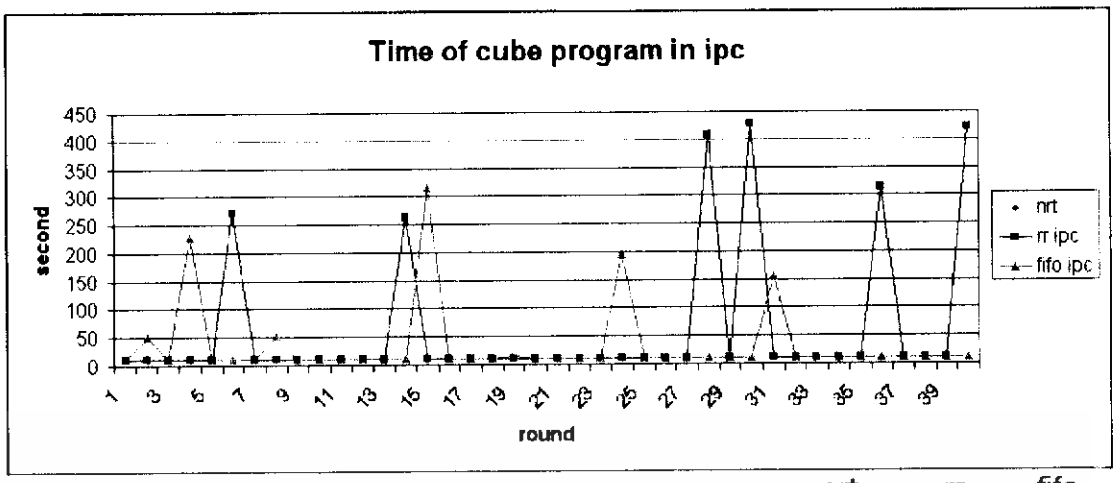
รูปที่ 4.3 กราฟแสดงเวลาที่ใช้ทำงานจนเสร็จของโปรแกรมที่ทำงานด้วยการวนรอบ 3 ชั้นด้วยรอบที่เท่ากัน(Cube program) โดยไม่มีการสื่อสารระหว่างกระบวนการ (Interprocess Communication, IPC)



	nrt	rr	fifo
averag	43.02	45	45.17
sd	0.15	0	0.38
mode	43	45	45
mean	43	45	45

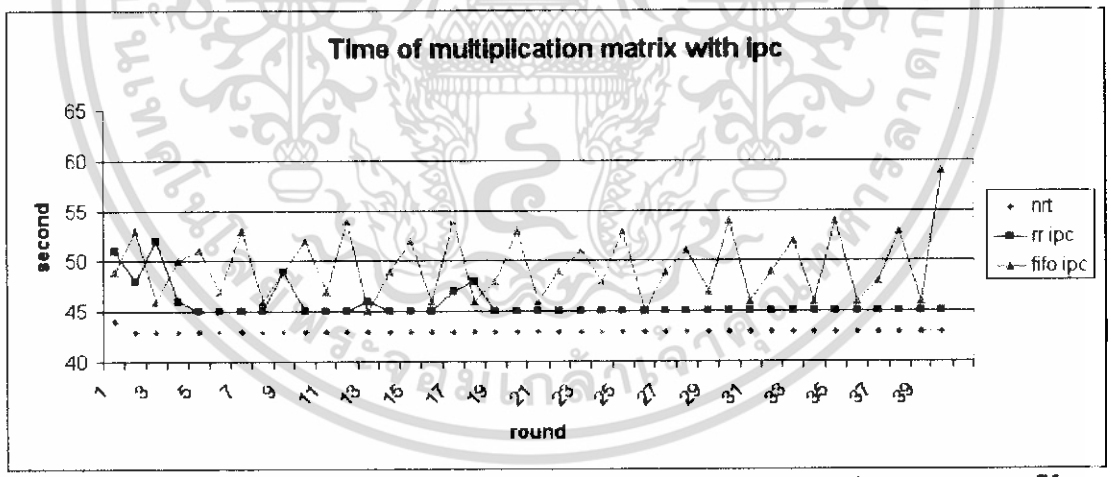
รูปที่ 4.4 กราฟแสดงเวลาที่ใช้ทำงานจนเสร็จของโปรแกรมการคูณ Matrix (Multiplication Matrix program) โดยไม่มีการสื่อสารระหว่างกระบวนการ (Interprocess Communication, IPC)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



	nrt	rr	fifo
average	11	62.85	35.22
sd	0	125.6	66.95
mode	11	12	12
mean	11	12	12

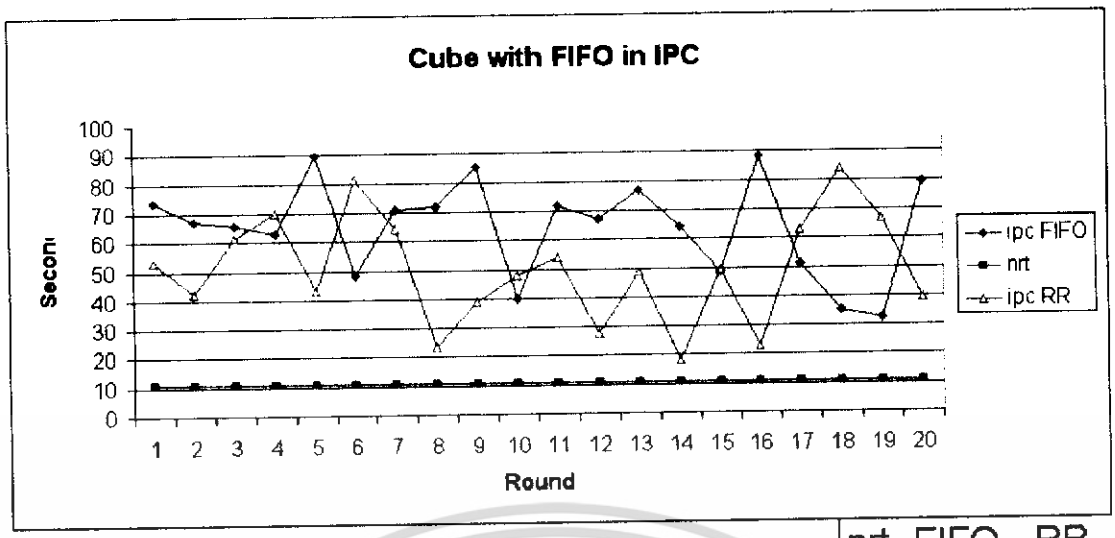
รูปที่ 4.5 กราฟแสดงเวลาที่ใช้ทำงานจนเสร็จของโปรแกรมที่ทำงานด้วยการวนรอบ 3 ชั้นด้วยรอบที่เท่ากัน(Cube program) โดยมีการสื่อสารระหว่างกระบวนการ (Interprocess Communication, IPC)



	nrt	rr	fifo
average	43.02	45.67	49.55
sd	0.15	1.65	3.34
mode	43	45	46
mean	43	45	49

รูปที่ 4.6 กราฟแสดงเวลาที่ใช้ทำงานจนเสร็จของโปรแกรมการคูณ Matrix (Multiplication Matrix program) โดยมีการสื่อสารระหว่างกระบวนการ (Interprocess Communication, IPC)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



	nrt	FIFO	RR
average	11	64.56	50
sd	0	16.9	18.6
mode	11	N/A	N/A
mean	11	67.16	49.1

รูปที่ 4.7 กราฟแสดงเวลาที่ใช้ทำงานจนเสร็จของโปรแกรมที่ทำงานด้วยการวนรอบ 3 ชั้นด้วยจำนวนรอบที่เท่ากัน(Cube Program) โดยมีการสื่อสารระหว่างกระบวนการ (Interprocess Communication, IPC) เมื่อทุกจุดมาจากค่าเฉลี่ยของเวลาในการทำงาน 24 ครั้ง

จากหลักการของการทำงานแบบเรียลไทม์ เราคาดว่าเวลาที่ใช้ของ โปรแกรมคำนวณทางคณิตศาสตร์(Cube Program และ Multiplication Matrix program) จะประเมินไม่ได้เมื่อมีการทำงานของโปรแกรมแบบเรียลไทม์ทำงานพร้อมกัน (คาดเดาไม่ได้ว่าจะเสร็จช้ากว่าเดิมเท่าไหร่ในแต่ละรอบ เนื่องจากเมื่อมีงานที่ทำงานแบบเรียลไทม์แล้วงานที่ไม่ได้เป็นเรียลไทม์จะถูกพักงานไว้)

จากรูปที่ 4.3 และ 4.4 จะเห็นได้ว่าทั้ง 2 โปรแกรมจะมีความสัมพันธ์กันในตัวโปรแกรม (เป็นการวนรอบ 3 ชั้นในจำนวนรอบที่เท่ากัน) แต่จะมีขั้นตอนการทำงานที่แตกต่างกันโดยโปรแกรมการคูณ Matrix จะมีการทำงานที่ซับซ้อน

เมื่อนำผลของกราฟทั้ง 2 มาเปรียบเทียบกันจะเห็นได้ว่าในรูปที่ 4.3 และ 4.4 นั้นเวลาในการทำงานของแต่ละโปรแกรมที่ไว้ใช้ทดสอบ เมื่อไม่มีการทำงานใด ๆ ของโปรแกรมที่มีการทำงานแบบเรียลไทม์จะใช้เวลาในการทำงานเฉลี่ยที่ต่ำกว่า (Cube program ใช้เวลา 11 วินาทีและ Matrix program ใช้เวลา 43.02 วินาที)

และเมื่อมีโปรแกรมที่ทำงานแบบเรียลไทม์ทำงานพร้อมกับโปรแกรมคำนวณทางคณิตศาสตร์ในแต่ละการจัดลำดับการทำงานจะมีค่าเวลาเฉลี่ยที่แตกต่างกันโดยส่วนใหญ่ค่าที่ได้จากการจัดลำดับการทำงานแบบมาก่อนได้ก่อน (First-Come, First-Served Scheduling) จะใช้เวลาโดยเฉลี่ย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มากกว่า การจัดลำดับการทำงานแบบเวียนเทียน (Round-Robin Scheduling) เพียงเล็กน้อย และผลเวลาที่คาดไว้ในตอนแรกคือน่าจะมีการกระโดดของเวลาให้เห็นได้ชัดในแต่ละรอบการทำงานซึ่งผลที่ได้ออกมานั้นจึงเป็นสิ่งที่เหนือความคาดหมายโดยที่ค่าเวลาไม่มีการเปลี่ยนแปลงมากนักในแต่ละรอบ (สังเกตได้จากค่าความเบี่ยงเบนมาตรฐาน (SD) มีค่าเพียงเล็กน้อย) กล่าวได้ว่ามีผลต่องานที่เป็นแบช(Batch)น้อยมาก ซึ่งค่าเวลาจากรูปที่ 4.3 และ 4.4 สามารถนำมาเปรียบเทียบเป็นตารางได้ดังนี้

	FIFO		RR	
	Average	**SD**	**Average**	**SD**
Cube program	12.7	0.72	12	0
Matrix program	45.17	0.38	45	0

ตารางที่ 4.1 ตารางแสดงการเปรียบเทียบของค่าเฉลี่ยและค่าความเบี่ยงเบนมาตรฐานของแต่ละการจัดลำดับการทำงาน(Scheduling) ในแต่ละโปรแกรม

จากรูปที่ 4.5 และ 4.6 ได้ให้โปรแกรมคำนวณทางคณิตศาสตร์ทำงานเหมือนในตอนแรก แต่จะมีโปรแกรมที่ไม่เป็นเรียลไทม์ติดต่อสื่อสารกับโปรแกรมที่เป็นโปรแกรมแบบเรียลไทม์โดยผ่านการสื่อสารระหว่างกระบวนการ(Interprocess Communication, IPC)เข้ามาทำงานพร้อมกัน และจากที่ได้ทำการทดสอบพบว่าเมื่อมีการติดต่อสื่อสารกันระหว่างงานที่เป็นแบบเรียลไทม์และงานที่ไม่เป็นเรียลไทม์ทั้งค่าที่ได้จากการจัดลำดับการทำงานแบบการจัดลำดับการทำงานแบบมาก่อนได้ก่อน (First-Come, First-Served Scheduling) และการจัดลำดับการทำงานแบบเวียนเทียน (Round-Robin Scheduling) จะทำให้เกิดการขัดจังหวะที่สังเกตได้ชัดจากเวลาในการทำงานของโปรแกรมคำนวณทางคณิตศาสตร์ซึ่งเราไม่สามารถคาดเดาได้ว่าโปรแกรมคำนวณทางคณิตศาสตร์เหล่านั้นจะทำงานจนเสร็จเมื่อไหร่ ซึ่งในที่นี้ได้มีการใช้เวลาในการทำงานนานกว่าเวลาในการทำงานปกติมากโดยที่ดูได้จากค่าเฉลี่ยของเวลาการทำงาน และจากการที่มีการทำงานของกระบวนการติดต่อสื่อสารระหว่างกระบวนการนั้นเองทำให้เราไม่สามารถคาดเดาได้ว่างานจะทำงานเสร็จเมื่อไหร่ซึ่งดูได้จากค่า SD เพราะฉะนั้นจากรูปที่ 4.5 และ 4.6 สามารถนำมาเปรียบเทียบเป็นตารางได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	SD with FIFO Scheduling		SD with RR Scheduling	
	Non IPC	IPC	Non IPC	IPC
Cube program	0.72	66.95	0	125.62
Matrix program	0.38	3.34	0	1.65

ตารางที่ 4.2 ตารางแสดงการเปรียบเทียบระหว่างค่าความเบี่ยงเบนมาตรฐานเมื่อมีและไม่มีการติดต่อสื่อสารระหว่างกระบวนการในแต่ละการจัดลำดับการทำงาน(Scheduling)

และจากผลการทดลอง 40 รอบในแต่ละการจัดลำดับการทำงานและการติดต่อสื่อสารระหว่างกระบวนการจะสังเกตว่ามีการกระโดดของค่าเวลา ดังนั้นทางผู้จัดทำได้ทำการทดลองเพิ่มเติมโดยการใช้การติดต่อสื่อสารระหว่างกระบวนการและใช้การจัดลำดับการทำงานทั้ง 2 แบบข้างต้น โดยที่ในแต่ละจุดจะใช้ผลเฉลี่ยของเวลาจากการทดลอง 24 รอบ (การทำงานทั้งหมดคือ 520 รอบและไม่นำผลในรอบที่ 1 ถึงรอบที่ 40 มาคำนวณซึ่งผลที่ได้ตามรูปที่ 4.7) ซึ่งผลที่ได้ออกมานั้น แม้ว่าจะใช้เวลาเฉลี่ยของจำนวนรอบที่เพิ่มขึ้นแล้วก็ตาม ก็ยังเห็นความไม่สม่ำเสมอของเวลาเฉลี่ยที่ใช้ทำงานจนเสร็จเหมือนเดิม ซึ่งเหตุผลของผลการทดลองที่เกิดขึ้นนี้น่าจะเกิดมาจากกลไกการทำงานภายในของตัวอัลกอริทึมของการติดต่อสื่อสารระหว่างกระบวนการนั่นเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปและเสนอแนะ

5.1 สรุปผลการทดลอง

จากการทดลองในโครงงานนี้ เริ่มจากการศึกษาทางทฤษฎีในด้านต่าง ๆ เกี่ยวกับระบบปฏิบัติการลินุกซ์ที่ทำงานแบบเรียลไทม์ อีกทั้งด้านการติดตั้งและการนำไปใช้ โดยเมื่อทำการติดตั้งเคอร์เนลของลินุกซ์เรียบร้อยจึงทำการติดตั้งเคอร์เนลพิเศษเพื่อที่จะทำให้ลินุกซ์มีความสามารถทำงานที่เป็นแบบฮาร์ดเรียลไทม์ได้ จากนั้นเตรียมสายเพื่อเชื่อมต่อพอร์ตอนุกรมและเขียนโปรแกรมติดต่อสื่อสารกันระหว่างเครื่องคอมพิวเตอร์ 2 เครื่องผ่านทางพอร์ตอนุกรมซึ่งโปรแกรมนี้เองถูกเขียนให้เป็นงานที่ทำงานแบบเรียลไทม์ อีกทั้งเขียนโปรแกรมคำนวณทางคณิตศาสตร์เพื่อที่จะเก็บค่าเวลาที่ทำงานจนเสร็จโดยโปรแกรมนี้ไม่ได้เป็นแบบเรียลไทม์ ซึ่งเมื่อทดลองเสร็จได้นำเอาเวลาที่ได้นี้มาวิเคราะห์ว่าพฤติกรรมการทำงานของงานที่เป็นแบบเรียลไทม์นั้นมีผลกระทบต่อโปรแกรมอื่น ๆ ที่ทำงานด้วยอย่างไร

ในที่นี้ได้ทดสอบแล้วว่าค่าที่ได้ทำการส่งจากเครื่องคอมพิวเตอร์เครื่องส่งไปยังคอมพิวเตอร์เครื่องรับนั้น สามารถรับได้ครบถ้วนและถูกต้องจริง ผลที่ได้สามารถดูได้จากรูปผลการทดลองในบทที่ 4 และในด้านของผลเวลาที่ได้จากการทดลองสามารถบ่งบอกได้ว่า เมื่อมีงานที่ทำงานแบบเรียลไทม์ทำงานพร้อมกับงานอื่น ๆ ที่ไม่ได้เป็นงานที่ทำงานแบบเรียลไทม์จะทำให้เกิดการขัดจังหวะการทำงาน ซึ่งส่งผลให้งานที่ไม่ได้เป็นแบบเรียลไทม์จะมีเวลาในการทำงานจนเสร็จมากกว่าปกติ โดยที่ในตอนแรกจะเห็นไม่ชัดนัก แต่หากมีการติดต่อสื่อสารระหว่างกระบวนการของงานที่เป็นแบบเรียลไทม์และไม่เป็นแบบเรียลไทม์(Interprocess Communication, IPC ดังรูปที่ 4.5 และ รูปที่ 4.6) จะเห็นได้ว่า มีการขัดจังหวะการทำงานของงานที่ไม่เป็นแบบเรียลไทม์ซึ่งทำให้งานถูกพักไว้และเวลาที่ทำงานจนเสร็จจะมากกว่าปกติมาก (ในกรณีที่ไม่มีการติดต่อสื่อสารระหว่างกระบวนการ)

ทั้งนี้เวลาที่ใช้ในการทำงานของโปรแกรม Cube.c มีค่าไม่คงที่เมื่อมีงานที่มีการติดต่อสื่อสารระหว่างกระบวนการกับงานที่เป็นแบบเรียลไทม์มากกว่าโปรแกรม Matrix.c น่าจะเกิดมาจากการที่โปรแกรม Cube.c นั้นมีการใช้งานหน่วยประมวลผลกลางเพียงอย่างเดียว(CPU Intensive) ซึ่งเมื่อถูกพักงานเนื่องจากการทำงานของโปรแกรมที่เป็นแบบเรียลไทม์จะต้องรอจนกว่างานที่เป็นแบบเรียลไทม์เสร็จสิ้นจึงสามารถทำงานต่อได้จึงทำให้จะเสียเวลาในการทำงานมาก แต่ถ้าสังเกตดูที่โปรแกรม Matrix.c นอกจากจะมีการทำงานของหน่วยประมวลผลกลางแล้วยังมีการทำงานของการสลับหน้าของหน่วยความจำหลัก(Memory)อีกด้วย ดังนั้นเมื่อมีการพักงานจากการทำงานของโปรแกรมที่เป็นแบบเรียลไทม์ตัวโปรแกรม Matrix.c ก็จะยังทำงานในส่วนที่เป็นการสลับหน้าของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน่วยความจำหลักต่อไป ซึ่งทำให้เวลาที่ใช้ในการทำงานนั้นไม่มีผลกระทบมากนัก โดยผลสามารถดูได้จากบทที่ 4

5.2 ปัญหาที่พบ

1. แหล่งข้อมูลที่ใช้ในการศึกษาเกี่ยวกับการทำงานแบบเรียลไทม์(Real Time System) ซึ่งในที่นี้ใช้ อาร์ทีลินุกซ์ มีน้อย ส่วนมากจะต้องหาจากทางอินเตอร์เน็ต และไม่มีความหลากหลายของเนื้อหา เนื่องจากแหล่งข้อมูลหลัก ๆ ได้อ้างอิงมาจากทางผู้พัฒนา อาร์ทีลินุกซ์ หรือก็คือ FSMLabs

2. จากการใช้งานของผู้ใช้หลายๆคน ได้กล่าวไว้ว่า "ในบางระบบ อาร์ทีลินุกซ์ ยังไม่มีความเสถียรภาพ" ซึ่งอาจจะมาจาก 2 สาเหตุคือ

2.1 เกิดจากการที่ระบุค่าต่าง ๆ ของเคอร์เนลไม่ตรงกับฮาร์ดแวร์ที่ใช้อยู่ ในบางกรณีถึงแม้ว่าจะทำการระบุค่าตรงกับฮาร์ดแวร์แล้วแต่ก็ไม่สามารถที่จะทำงานได้ เช่น เมื่อทำการติดตั้งเคอร์เนลต้องระบุประเภทของหน่วยประมวลผลกลาง ซึ่งในที่นี้ระบบใช้หน่วยประมวลผลกลางของค่ายอินเทล รุ่นเพนเทียม 4 แต่ว่าต้องระบุค่าเป็น 386/486/586/686 เพื่อที่จะให้ระบบสามารถทำงานได้

2.2 เกิดจากข้อผิดพลาดจากโปรแกรมที่ทำงาน เนื่องจากการเขียนโปรแกรมแบบเรียลไทม์นี้ยังไม่มีมาตรฐานในการเขียน ซึ่งจากผู้ที่ใช้งานหลาย ๆ คนกล่าวไว้ว่า "ในบางครั้งได้พบปัญหาที่ว่าไม่สามารถรันโปรแกรมแบบเรียลไทม์ได้ ซึ่งสาเหตุน่าจะเกิดมาจากวิธีการเขียนโปรแกรมแบบเรียลไทม์นี้เอง"

3. ปัญหาอื่น ๆ สามารถหาได้จาก http://hq.fsmlabs.com/mailman/listinfo/rtl และ http://www2.fsmlabs.com/mailing_list/rtl.w5archive/9903/index.html ซึ่งเป็นที่รวมข้อสงสัยหรือปัญหาต่าง ๆ ของอาร์ทีลินุกซ์

5.3 ข้อเสนอแนะ

1. ในการระบุการจัดลำดับของการทำงาน(Scheduling) มีบางวิธีที่ไม่ได้นำมาทดลอง เช่น SCHED_OTHER เนื่องจากอยู่นอกเหนือขอบเขตที่ต้องการ

2. มีการจัดลำดับของการทำงาน (Scheduling) ที่ถูกพัฒนาขึ้นหลังจากมีการให้ค่าความสำคัญแบบคงที่ (Fix Priority) เช่น EDF ซึ่งถ้าต้องการที่จะใช้การจัดลำดับของการทำงาน (Scheduling) จำพวกนี้จะต้องทำการติดตั้งเคอร์เนลเสริมเสียก่อน เนื่องจากอยู่นอกเหนือขอบเขตที่ต้องการทางผู้จัดทำจึงไม่ได้ทำการทดสอบด้วย การจัดลำดับของการทำงาน (Scheduling) นี้

3. ในการสื่อสารระหว่างกระบวนการ(Interprocess Communication, IPC) ที่ทางอาร์ทีลินุกซ์ได้จัดเตรียมไว้ให้ยังมีอีก 1 วิธีคือการใช้หน่วยความจำร่วม (Share Memory) แต่ที่เลือกใช้ แบบ FIFO เพราะว่าเป็นวิธีที่สามารถเข้าใจและทำการเขียนโปรแกรมได้ง่าย อีกทั้งยังเป็นที่นิยมอีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. *โครงงานนี้ได้ใช้โปรแกรมที่เป็นแบบเรียลไทม์ในส่วนของการรับและส่งข้อมูลผ่านทางพอร์ทอนุกรม* ซึ่งได้ทำการวัดที่ฝั่งรับเพียงอย่างเดียวเนื่องจากเป็นการศึกษาเกี่ยวกับระบบการทำงานจริงบนระบบปฏิบัติการลินุกซ์โดยผ่านพอร์ทอนุกรม *อีกทั้งในด้านฝั่งส่งและรับได้ใช้*โปรแกรมที่ทำให้เกิดงานที่เป็นแบบเรียลไทม์เพียง 1 งานเท่านั้น ดังนั้นโครงงานนี้สามารถพัฒนาในการศึกษาไปได้อีกเช่น เพิ่มให้งานที่เป็นแบบเรียลไทม์ทั้งทางด้านรับหรือส่งมีมากกว่า 1 งานขึ้นไปให้งานนั้นๆ แย่งกันทำงานแล้วศึกษาผลที่เกิด เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] อานันท์ สีห์พิทักษ์เกียรติ. **UNIX การใช้งานเบื้องต้น**. เชียงใหม่ : มหาวิทยาลัยเชียงใหม่.

[2] A. Castellani, et. al. 2547. **Advanced Teleoperation Architecture.Workshop at IEEE International Conference on Robotics and Automation (ICRA)**. Barcellona.

[3] Chang-Gun Lee. **RTLinux installation Guide.** [Online]. Available : http://www.ece.osu.edu/~cglee/EE694Z/rtlinux/rtlinuxInstallation2005Sp.pdf.

[4] **How to Install RTLinux (rtlinux-3.2-pre1) with RedHat 7.3 (linux-2.4.18-3).** [Online]. Available : http://www.csse.monash.edu.au/~sislam/FAQS/How_to_Install_RTLinux.html.

[5] Ismael Ripoll. 2540. **Real-Time Linux (RT-Linux)**. [Online]. Available : http://www.tldp.org/linuxfocus/English/May1998/article44.html.

[6] Ismael Ripoll. 2544. **RTLinux versus RTAI. 2002.** [Online]. Available : http://bernia.disca.upv.es/rtportal/comparative/rtl_vs_rtai.html

[7] Kenneth J. Hendrickson. **Report on the 2nd Real-Time Linux Workshop.** [Online]. Available : http://www.mlinux.org/members/ken/

[8] M. Barabanov. 2539. **"A Linux-based RealTime Operating System."** Master's thesis, New Mexico Institute of Mining and Technology

[9] Michael R.Sweet. **Serial Programming Guide for POSIX Operation System**. [Online]. Available : http://www.easysw.com/~mike/serial/serial.html.

[10] Michael Barabanov and Victor Yodaiken. 2538. "Real Time Linux." **Linux Journal**. 1997.

[11] Nicholas Mc Guire. **MINIRTL – HARD REAL TIME LINUX FOR EMBEDDED SYSTEM**. [Online]. Available : http://www.linuxdevices.com/files/article005/p-c08_guire.pdf.

[12] Riku Saikkonen. 2542. **Linux I/O port programming mini-HOWTO.** [Online]. Available : http://www.tldp.org/HOWTO/IO-Port-Programming.html.

[13] T. Heimfarth, M. Götz, F. J. Rammig, F.R. Wagner. 2545. **RTC: A Real-time Communication Middleware on Top of RTAI-Linux**. IEEE ISORC'03. Japan.

[14] Taneli V¨ah¨akangas. **Real-Time Issues in Linux.** [Online]. Available : http://www.cs.helsinki.fi/u/kraatika/Courses/sem01a/vahakangas.pdf.

[15] Toshikazu Ohkubo, Tatsuo Takahashi, Shunji Futagami and Kazufumi

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก.

Code

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Cube.c

```
1.  #include <time.h>
2.  main(int argc,char *arg[])
3.  {
4       double a;
5       clock_t start,end;
6       int c; double t[40];
7       int i,j,k,m; double temp = 1;
8       for(m = 1;m<=40;m++)
9       {
10          temp = 0;
11          start = time(NULL);
12          for (i=0;i<1000;i++)
13          {
14              for (j=0;j<1000;j++)
15              {
16                  for (k=0;k<1000;k++)
17                  {
18                  temp++;
19                  a = i*j*k;
20                  }
21              }
22          }
23          end =time(NULL);
24          t[m-1] = (double)((end-start)/1000);
25          printf("round = >> %d << and time >> %f <<\n",m,t[m-1]);
26      }
27  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Matrix.c

```
1.  #include <time.h>
2.  int m,n,p,q,k;
3.  int i;int j;
4   double A[1000][1000],B[1000][1000];
5   double Ans[1000][1000];
6   clock_t start,end;
7   double t[40];
8   main (int argc,char *arg[])
9   {
10          m = 1000; // row of A
11          n = 1000; // coloumn of A
12          p = 1000; // row of B
13          q = 1000; // coloumn of B
14          int r;
15          for (r = 1;r <= 40;r++)
16          {
17              for (i = 1; i <= m ; i++)
18              {
19                  for (j = 1 ; j <= n ; j++)
20                  {
21                      A[i-1][j-1] = 10;
22                  }
23              }
24              for (i = 1; i <= p ; i++)
25              {
26                  for (j = 1 ; j <= q ; j++)
27                  {
28                      B[i-1][j-1] = 10;
29                  }
30              }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Matrix.c (ต่อ)

```
31                  start = time(NULL);
32                  for (i = 0; i < m ; i++)
33                  {
34                          for (j = 0 ; j < q ; j++)
35                          {
36                                  for (k=0;k < n;k++)
37                                  {
38                                          Ans[i][j] = Ans[i][j] + ((A[i][k])*(B[k][j]));
39                                  }
40                          }
41                  }
42                  end =time(NULL);
43                  t[r-1]=(double)((end-start)/1000);
44                  printf("round = >> %d << and time >> %f <<\n",r,t[r-1]);
45          }
46      return 0;
47  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sent.c

```
1.  #include <rtl.h>
2.  #include <time.h>
3.  #include <pthread.h>
4   #include "/usr/src/rt_com/rt_com.h"
5   pthread_t thread;
6   void * start_routine(void *arg)
7   {
8           int n;
9           char buf[1];
10          struct sched_param p;
11          p . sched_priority = 1;
12          pthread_setschedparam (pthread_self(), SCHED_RR, &p);
13          pthread_make_periodic_np (pthread_self(), gethrtime(), 100000000);
14          int i=0;
15          char hello[] = "hello";char tmp;
16          while (1) {
17              pthread_wait_np ();
18              i++;
19              tmp = hello[i % 5];
20              rt_com_write( 0, &tmp , 1 );
21              rtl_printf( ">> %c << sent. and round is >> %d <<\n",tmp,(int)i);
22          }
23          return 0;
24  }
25  int init_module(void) {
26          rt_com_setup(0, 38400, RT_COM_PARITY_NONE, 1, 8);
27          return pthread_create (&thread, NULL, start_routine, 0);
28  }
29  void cleanup_module(void) {
30          pthread_delete_np (thread);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sent.c (ต่อ)

```
31                  printk("finish sent task.\n");
32                  rt_com_setup(0, -1, 0, 0, 0);
33          }
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Makefile(sent.c)

```
1.      include /usr/rtlinux/rtl.mk
2.      all:    sent.c
3               S(CC) `rtl-config --cflags` -Wall -c sent.c -o sent.o
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Makefile(receive.c)

```
1.      include /usr/rtlinux/rtl.mk
2.      all:    receive.c
3               $(CC) `rtl-config --cflags` -Wall -c receive.c -o receive.o
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Makefile(rtfifo.c)

```
1.  include /usr/rtlinux/rtl.mk
2.  CC=gcc
3.  all: rtfifo.c rtfifo.o
4   batch: batch.c common.h
5           S{CC} `rtl-config --include` -Wall -O2 batch.c -o batch
6   rtfifo.o: rtfifo.c common.h
7           ${CC} `rtl-config --cflags` -c rtfifo.c -o rtfifo.o
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Receive.c

```
1.  #include <rtl.h>
2.  #include <time.h>
3.  #include <pthread.h>
4   #include "/usr/src/rt_com/rt_com.h"
5   pthread_t thread;
6   char buf[1];
7   void * start_routine(void *arg)
8   {
9           struct sched_param p;
10          p . sched_priority = 1;
11          pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);
12          pthread_make_periodic_np (pthread_self(), gethrtime(), 100000000);
13          int i = 0;
14          do {
15              int n;
16              pthread_wait_np();
17              n = rt_com_read(0, buf, sizeof(buf));
18              if (n > 0) {
19                  i++;
20                  rtl_printf("round : >> %d << and data is >> %s\n",i,buf);
21              }
22          } while (1);
23          return 0;
24  }
25  int init_module(void) {
26          rt_com_setup(0, 38400, RT_COM_PARITY_NONE, 1, 8);
27          return pthread_create (&thread, NULL, start_routine, 0);
28  }
29  void cleanup_module(void) {
30          pthread_delete_np (thread);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Receive.c (ต่อ)

```
31                  printk("finish sent task.\n");
32                  rt_com_setup(0, -1, 0, 0, 0);
33          }
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Batch.c

```
1.  #include <stdio.h>
2.  #include <fcntl.h>
3.  #include <unistd.h>
4   #include <rtl_time.h>
5   #include "common.h"
6   #include <time.h>
7   int main( void )
8   {
9       int fd_data, i;
10      double t,to;
11      struct timeStamp data;
12      if ((fd_data = open("/dev/rtf1", O_RDONLY)) < 0) {
13          printf("Error opening /dev/rtf1 \n");
14          return 0;
15      }
16      for (i = 0; i < 500000 ; i++) {
17          read(fd_data, &data, sizeof(data));
18          to = time(NULL);
19          t = to-data.timeS;
20          printf("data.timeS = %l \tdifferent = %f. \t round =
21  %d\n",data.timeS,t,i );
22      }
23      close( fd_data );
24      return 0;
    }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Common.h

```
1.      #define DATA_FIFO 1
2.      #define DATA_FIFO_SIZE 4000
3.      struct timeStamp{
4       double timeS;
5       };
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Rtfifo.c

```
1.      #include <rtl.h>
2.      #include <pthread.h>
3.      #include <rtl_fifo.h>
4       #include "common.h"
5       #include <time.h>
6       #include "/usr/src/rt_com/rt_com.h"
7       char buf[1];
8       struct timeStamp data;
9       pthread_t periodic_thread;
10      void * periodic_function( void * arg )
11      {
12              struct sched_param p;
13              p . sched_priority = 1;
14              pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);
15              pthread_make_periodic_np (pthread_self(), gethrtime(), 100000000);
16              int i = 0;  int n;
17              do {
18                      pthread_wait_np();
19                      n = rt_com_read(0, buf, sizeof(buf));
20                      data.timeS = time(NULL);
21                      if (n > 0) {
22                              i++;
23                              rtf_put( DATA_FIFO, &data, sizeof(data) );
24                              rtl_printf("round : >> %d << and data is >> %s\n",i,buf);
25                      }
26              } while(1);
27              return 0;
28      }
29      int init_module(void)
30      {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Rtfifo.c (ต่อ)

```
31              rtf_create( DATA_FIFO, DATA_FIFO_SIZE );
32              rt_com_setup(0, 38400, RT_COM_PARITY_NONE, 1, 8);
33              return pthread_create( &periodic_thread, NULL, periodic_function, 0 );
34      }
35      void cleanup_module(void)
36      {
37              rtf_destroy( DATA_FIFO );
38              pthread_delete_np( periodic_thread );
39              rt_com_setup(0, -1, 0, 0, 0);
40              printk("Bye\n");
41      }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข.

ระบบปฏิบัติการแบบเรียลไทม์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คณะผู้จัดทำได้กล่าวถึงระบบปฏิบัติการแบบเรียลไทม์ที่นิยมใช้ไว้ในบทที่ 2 ภาคผนวกนี้จะนำเสนอระบบปฏิบัติแบบเรียลไทม์อื่นๆที่มีใช้กันในปัจจุบัน แสดงดังตารางที่ ข-1

Operating System Name	Manufacture
Arx RTOS	Seoul National University
AvSYS	Avocet Systems
CMX RTOS	CMX Systems
Chorus	Jaluna S.A.
EROS	U Penn
Harmony	National Research Council
Intergrity	Green Hills Software
iRMX	TenAsys
LynxOS	Lynuxworks
Maruti	University of Maryland
Nucleus	Mentor Graphics
OS-9	Microwave
Oncore OS	Oncore Systems Corporation
On Time RTOS-32	On Time
OSE RTOS	ENEA
PDOS	EYRING corporation
pSOSystem 3	Wind River
QNX Neutrino RTOS	QNX Software System Ltd
Precise/MQX RTOS	ARC International
Real Time Mach	CMU
REAL/IX PX	Modoomp
RTMX O/S	RTMX Incorporated
SMX RTOS	Micro Digital Inc.
SUMO	Lancaster University
ThreadX	Green Hills Software

ตารางที่ ข-1 ระบบปฏิบัติการแบบเรียลไทม์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค.

การติดตั้งอาร์ทีลินุกซ์

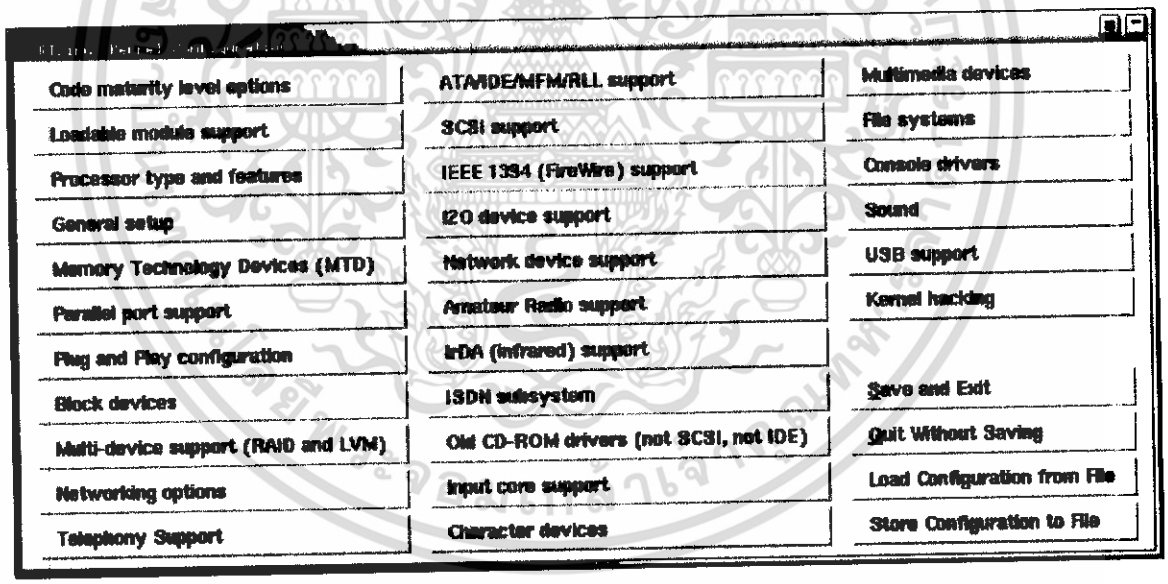
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนการติดตั้งอาร์ทีลินุกซ์โดยละเอียดมีดังนี้

1. rtlinux-3.1.tar.gz จาก http://www.fsmlabs.com/
2. linux-2.4.20.tar.gz จาก http://www.kernel.org/ เป็นเคอร์เนลที่ rtlinux 3.1 สนับสนุน

เมื่อได้ไฟล์ตามที่ต้องการมาแล้วให้ทำการคัดลอกไฟล์ที่ได้ไปที่ /usr/src/ แล้วสั่งคำสั่งต่อไปนี้

1	/usr/src > tar zxvf linux-2.4.20.tar.gz	คลายการบีบอัดไฟล์ linux-2.4.20.tar.gz
2	/usr/src > ln –s linux-2.4.20 linux	สร้างลิงค์ของไดเร็คตอรี linux-2.4.20
3	/usr/src > tar zxjf rtlinux-3.1.tar.gz	คลายการบีบอัดไฟล์ rtlinux-3.1.tar.gz
4	/usr/src > cd linux	เข้าไปในไดเร็คตอรี linux
5	/usr/src/linux > patch –p1 < ../rtlinux-3.1 /kernel_patch-2.4.20-rtl rtlinux-3.1	แก้ไขซอร์สโคดของเคอร์เนลด้วย patch ที่เตรียมไว้ให้
6	/usr/src/linux > make xconfig	คำสั่งในการปรับค่าเคอร์เนลจะได้ดังรูปที่ 3.1

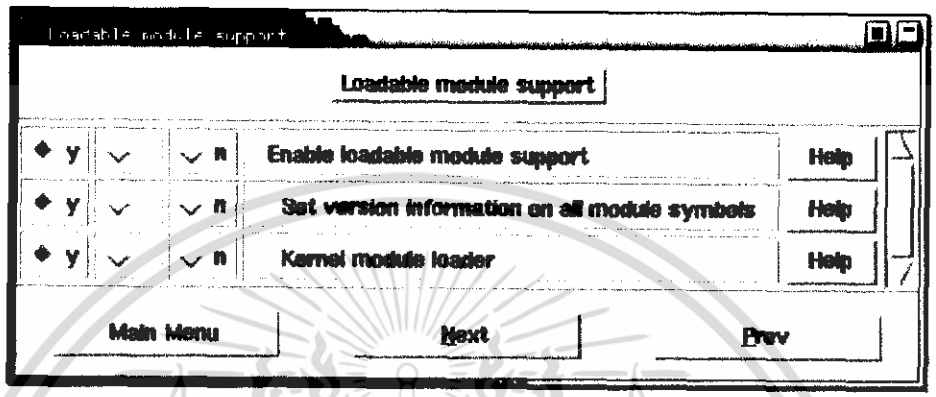


รูปที่ ค-1 หน้าจอหลักของการปรับค่าเคอร์เนล

ในการปรับค่าตัวเลือกของเคอร์เนลจะมีตัวเลือกให้เลือกมากมายในหลายๆ หมวดหมู่ ขอให้พยายามเลือกตัวเลือกต่างๆ ให้เหมาะกับคอมพิวเตอร์ที่เราใช้อยู่ให้มากที่สุด ตัวเลือกไหนไม่เข้าใจไม่แน่ใจ พยายามกดปุ่มขอความช่วยเหลือ ("Help") เพื่ออ่านดูคำแนะนำ โดยปกติตัวเลือกในแต่ละรายการจะมีทางเลือกให้ 3 ทางคือ เลือกให้อยู่ในเคอร์เนล(y), เลือกเป็นโมดูล(m) หรือ ไม่ต้องการ

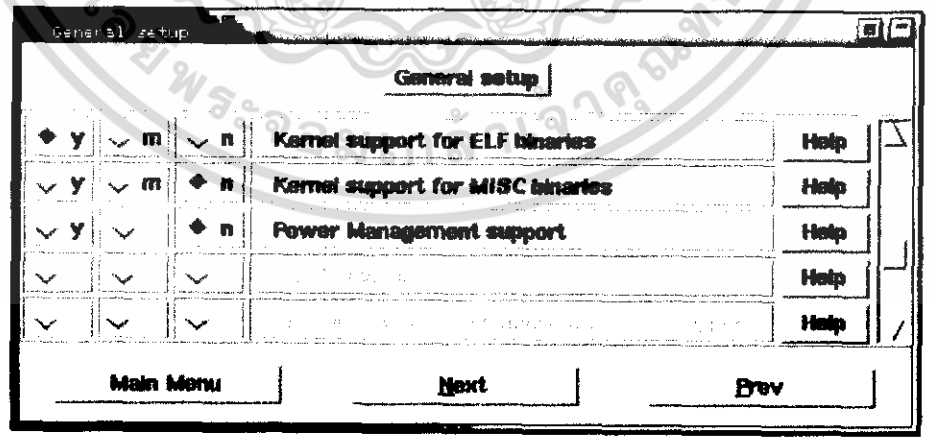
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(n) ในกรณีเลือกเป็นโมดูล หมายความว่าเมื่อจำเป็นต้องใช้ความสามารถนั้นๆ เราค่อยทำการแทรกโมดูลนั้นๆเข้าไปในเคอร์เนล วิธีนี้ทำให้ตัวเคอร์เนลมีขนาดเล็กและสามารถที่จะเพิ่มความสามารถในการทำงานเข้าไปได้เมื่อต้องการใช้งาน การจะเลือกให้เคอร์เนลสามารถแทรกโมดูลได้จะต้องเลือกตัวเลือกดังรูปที่ ค-2 ในหัวข้อ "Enable loadable modules support" ให้อยู่ในเคอร์เนล



รูปที่ ค-2 เลือกให้เคอร์เนลสามารถแทรกโมดูลได้

การปรับค่าตัวเลือกของเคอร์เนลสำหรับอาร์ทีลินุกซ์ มีข้อแนะนำในเอกสารของอาร์ทีลินุกซ์ อยู่ว่า อย่าเลือกให้เคอร์เนลสนับสนุนการทำงานของ APM BIOS เพราะไม่อาจคาดเดาผลกระทบที่จะเกิดขึ้นกับการทำงานเรียลไทม์ได้ ดังนั้นควรเลือกตัวเลือกดังรูปที่ ค-3 ในหัวข้อ "Power Management support" ไม่ให้อยู่ในเคอร์เนล



รูปที่ ค-3 ตัวเลือกทั่วไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากปรับค่าตัวเลือกต่างๆ เสร็จแล้ว ให้กดปุ่ม "Save and Exit" ที่หน้าหลักเพื่อเตรียมตัวคอมไพล์เคอร์เนลต่อไป แต่หากเราต้องการเก็บค่าตัวเลือกต่างๆ ที่เราได้ทำไว้เพื่อนำไปใช้ในเครื่องอื่น หรือ เพื่อเก็บไว้ใช้ในครั้งต่อๆ ไป ให้กดปุ่ม "Store Configuration to File" ที่หน้าหลัก เพื่อเก็บค่าตัวเลือกต่างๆ ลงแฟ้มข้อมูล เวลาที่เราจะนำค่าที่ได้เก็บไว้ไปใช้ สามารถทำได้โดยการกดปุ่ม "Load Configuration from File"

7	/usr/src/linux > make dep	ทำการตรวจสอบความเกี่ยวเนื่องกันของแต่ละไฟล์
8	/usr/src/linux > make bzImage	สร้างไฟล์สำหรับการบูต
9	/usr/src/linux > make modules	สร้างความสามารถที่เราเลือกให้เป็นโมดูล
10	/usr/src/linux > make modules_install	สำเนาไฟล์โมดูลต่างๆไปที่ /lib/modules/2.4.20-rtl
11	/usr/src/linux > cp arch/i386/boot/bzImage /boot/rtzImage32	สำเนาไฟล์สำหรับบูตไปที่ไดเร็กตอรี /boot

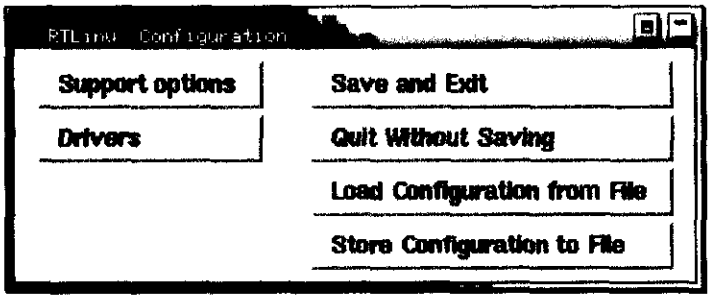
หลังจากนั้นให้ทำการแก้ไขไฟล์ของบูตโหลดเดอร์ในที่นี้ใช้ไลโล(LILO) ไฟล์นี้จะอยู่ที่ไดเร็กตอรี /etc/lilo.conf โดยให้เพิ่มข้อความต่อท้ายดังนี้

12	image = /boot/rtzImage32
13	label = rtlinux
14	read-only
15	root = /dev/hda2 (ให้ปรับตามพาร์ติชันของเครื่องผู้ใช้เอง)

ทำการบันทึกไฟล์ lilo.conf แล้วทำการรันคำสั่ง /sbin/lilo ที่เชลล์และรีบูตเครื่องหลังจากนั้นที่บูตโหลดเดอร์เราจะเห็นเคอร์เนลของอาร์ทีลินุกซ์ที่เราได้ทำการติดตั้งให้ทำการเลือก เคอร์เนลของอาร์ทีลินุกซ์ เมื่อบูตเครื่องขึ้นมาแล้วเราจะทำการติดตั้งโมดูลของอาร์ทีลินุกซ์เพื่อใช้ในการเพิ่มคุณสมบัติให้เคอร์เนลสามารถทำงานเรียลไทม์ได้โดยมีให้พิมพ์คำสั่งดังนี้

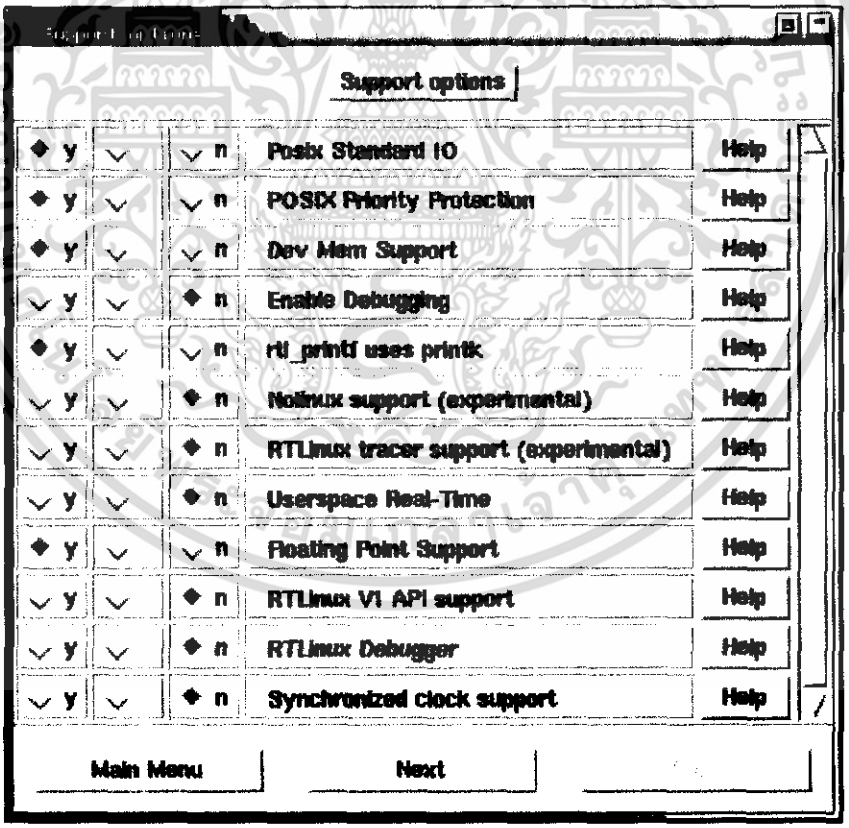
16	cd /usr/src/rtlinux	เข้าไปที่ไดเร็คตอรี rtlinux
17	/usr/src/rtlinux > ln - s ../linux linux	สร้างลิงค์ของไดเร็คตอรี่ linux
18	/usr/src/rtlinux > make xconfig	เรียกหน้าจอการปรับเคอร์เนลได้ดังรูปที่ ค-4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



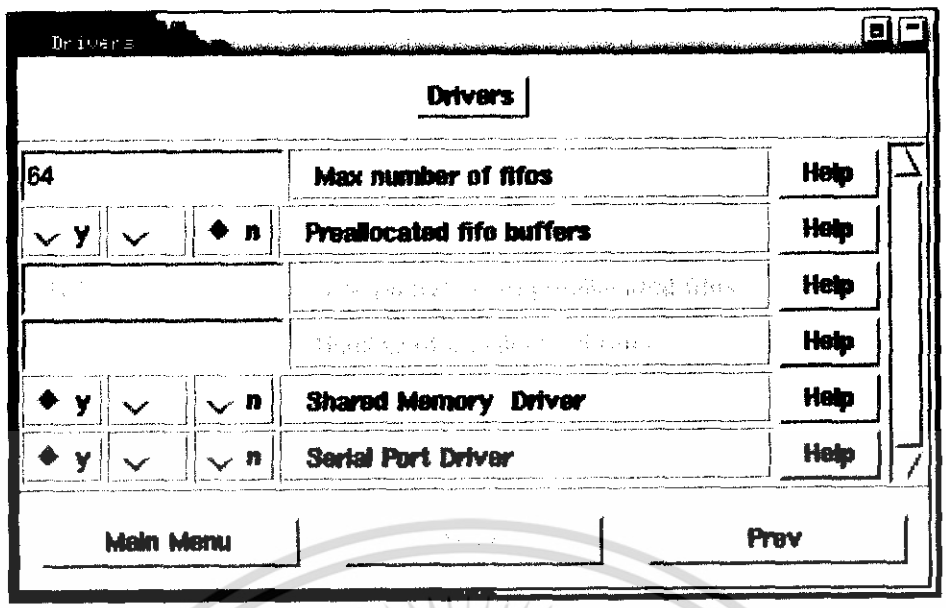
รูปที่ ค-4 หน้าจอการปรับเคอร์เนลของอาร์ทีลินุกซ์

ให้ทำการเลือกหัวข้อ "Support options" และทำการปรับค่าดังรูปที่ ค-5 เมื่อทำการปรับค่าต่างๆ แล้วให้ทำการเลือกหัวข้อ Drivers ดังรูปที่ ค-6 ซึ่งจะรวม Drivers ต่างๆ ที่มาพร้อมกับอาร์ทีลินุกซ์ โดย fifo และ shared memory จะใช้สำหรับส่งผ่านข้อมูลระหว่างโมดูลที่ทำงานเรียลไทม์(อยู่ในเคอร์เนลเสปซ) กับโปรแกรมที่ทำงานบนลินุกซ์(อยู่ในยูสเซอร์เสปซ) ความสามารถนี้จะเป็นประโยชน์อย่างมากเมื่อใช้พัฒนาโปรแกรมที่จะทำงานเรียลไทม์ เมื่อเลือกค่าต่างๆ แล้วให้กดปุ่ม "Save and Exit" ที่หน้าหลัก



รูปที่ ค-5 การปรับตัวเลือกใน Support options

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ค-6 การปรับค่าตัวเลือกใน Drivers

หลังจากนั้นพิมพ์คำสั่งดังนี้

19	/usr/src/rtlinux > make	ทำการคอมไพล์โมดูลต่างๆ
20	/usr/src/rtlinux > make install	ทำการติดตั้งโมดูล
21	/usr/src/rtlinux > make regression	คำสั่งในการทดสอบโมดูลพื้นฐาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Yamamoto. **A Benchmark Program and Normalization Method for Evaluating Communication System-Oriented Real-time OS Performace**. Tokyo.

[16] Alia K. Atlas, Azer Bestavros, "**Design and Implementation of Statistical Rate Monotonic Scheduling in KURT Linux**". Computer Science Department Boston University. Boston.

[17] W.Morven Gentleman . "**Multiprocesser Realtime Applications**". National Research Council of Cannada, Canada.

[18] Toshikazu Ohkubo, Ichiro Takenaka,Tetsuo Wasano, Masayuki Omiya. "**Quality Criteria for Realtime Microkernel Products**". NTT Software Co. Kurume Institute of Technology, ATR International, Oki Electric Industry *Co.*, Ltd.

[19] Aiguo He. "**RIDEE: a Realtime and Interactive Distance Education Support System**". Core and Information Technology Center, The University of Aizu.

[20] David Ingram. "**Soft Real Time Scheduling for General Purpose Client-Server Systems**". University of Cambridge Computer Laboratory Pembroke Street, Cambridge, CB2 3QG, United Kingdom.

[21] Cort Dougan. "**RTLinux and RTLinux/Pro in 2002**". Finite State Machine Labs, USA.

[22] Arnoldo D__az, Ismael Ripoll, Alfons Crespo and Patricia Balbastre. "**A New Application-De_ned Scheduling Implementation in RTLinux**". Universidad Polit_ecnica de Valencia Camino de Vera s/n, Valencia, Spain.

[23] Victor Yodaiken. "**The RTLinux Manifesto**". Department of Computer Science New Mexico Institute of Technology.

[24] Victor Yodaiken. "**FSMLabs RTLinux technology for hard Realtime**".

[25] Herman Bruyninckx. "**Real-Time and Embedded Guide**". Leuven,Belgium.

[26] FSM Labs,Inc. "**Getting Started with RTLinux**". April 20,2001.

[27] Patricia Balbastre, Ismael Ripoll. **Earliest Deadline First (EDF) Scheduler.** [Online]. Available : http://en.wikipedia.org/wiki/Scheduling.

[28] Wikipedia, the free encyclopedia. **Scheduling.** [Online]. Available : http://rtportal.upv.es/apps/edf-sched/index.shtml.

[29] Cort Dougan, Matt Sherer. "**RTLinux POSIX API for IO on Real-time FIFOs and Shared Memory".** [Online]. Available : www.fsmlabs.com/articles/arbfifo/arbfifo.pdf.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[30] Victor Yodaiken, Michael Barabanov. "**RTLinux Version Two**". [Online]. Available : http://rtportal.upv.es/tutorial/documentacion/design.pdf.

[31] MNET Lab, NTHU. 2002. "**RTLinux Driver Trace – rt_com**".

[32] Fred Proctor. "**EXAMPLE 4: FIFO COMMUNICATION BETWEEN RTL AND LINUX**". [Online]. Available : www.isd.mel.nist.gov/projects/ **rtl**inux/rtutorial-2.0/doc/ex0**4_fifo**.htm.

[33] Fred Proctor. "**Example 6: Shared Memory Communication Between RTL and Linux**". [Online]. Available : www.isd.mel.nist.gov/projects/ **rtl**inux/rtutorial-2.0/doc/ex0**6**_shm.htm.

[34] Andres Terrasa, Ana Garcia-Fornes, Agustin Espinosa. "**A Summary of the POSIX Trace Standard**". [Online]. Available : www.ocera.org/archive/upvlc/public/components/ptrace/ptrace.

[35] Andres Terrasa, Ana Garcia-Fornes, Agustin Espinosa. "**RTL POSIX Trace 1.0 (a POSIX Trace System in RT-Linux)**". [Online]. Available : www.ocera.org/archive/upvlc/public/components/ptrace/ptrace-1.0-1/doc/rtl-pt1.0.pdf.

[36] "**Execution Scheduling**". [Online]. Available : h30097.www3.hp.com/docs/posix/PCD1/DOCU_014.HTM.

[37] Harsha Perla, Veena Pai. "**Serial Communication via RS232 Port**". [Online]. Available : http://electrosofts.com/serial/index.html.

[38] "**SETSCHEDULER(2) Linux Programmer's Manual SETSCHEDULER(2)**". [Online]. Available : http://www.bigbiz.com/cgi-bin/manpage?2+sched_setscheduler.

[39] Matt Sherer. "**Writing Applications with RTLinux**". [Online]. Available : http://fsmlabs.com/developers/white_papers/emb_lj.htm. 2001.

[40] Richard Fitzpatrick. "**The Linux operating system**". [Online]. Available : http://farside.ph.utexas.edu/teaching/329/lectures/node8.html. 2003.

[41] FSMLabs Inc. "**RTLinux man page index**". [Online]. Available : http://cs.uccs.edu/~chow/pub/rtl/doc/html/MAN/rtl_index.4.html. 2001.

[42] aytak. "**How to Auto mount USB drive on Linux".** [Online]. Available : www.thailinuxcafe.com.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้