บทที่ 3

ระบบอุปกรณ์รับข้อมูลเข้าและส่งข้อมูลออกบนระบบยูนิกซ์

ในบทนี้จะขอกล่าวถึงการจัดการรับข้อมูลเข้า/ส่งออก (input/output) ในระบบปฏิบัติการยูนิกซ์พร้อมทั้งโปรแกรมย่อยควบคุมอุปกรณ์รับข้อมูลเข้า/แสดงผล โดยเน้นถึงการทำงานของโปรแกรมย่อยซึ่งใช้ควบคุมจอควบคุมเป็นหลัก

การจัดการกับอุปกรณ์รับข้อมูลเข้า/แสดงผลบนเครื่องคอมพิวเตอร์ต่างๆ นั้น มักจะแตกต่างกันไปตามระบบปฏิบัติการ เช่น บางระบบ มีโมดูลพิเศษที่อยู่นอกเคอร์เนล (kernel) ทำหน้าที่กำหนดและดูแลรักษาระบบแฟ้มข้อมูล แต่สำหรับยูนิกซ์การรับข้อมูลเข้า/ส่งข้อมูลออกบนแฟ้มหรือบนอุปกรณ์ต่างๆ นั้น กลับกลายเป็นหน้าที่ของเคอร์เนล การที่ออกแบบเช่นนี้ มีวัตถุประสงค์เพื่อให้ลดความแตกต่างที่เกิดขึ้นระหว่างการทำรับข้อมูลเข้า/แสดงผลบนอุปกรณ์ที่จัดเก็บแฟ้มข้อมูล กับการทำรับข้อมูลเข้า/แสดงผลบนอุปกรณ์ประเภทเทอร์มินัลและเครื่องพิมพ์ทำให้โปรแกรมของผู้ใช้ สามารถเรียกใช้อุปกรณ์ต่างๆ ในรูปแบบเดียวกัน

ระบบแฟ้ม (file system)

ระบบแฟ้มข้อมูลของยูนิกซ์นั้นประกอบด้วยแฟ้ม 3 ประเภท คือ

- แฟ้มธรรมดา (regular file)
- สารบบ (directory)
- แฟ้มพิเศษ (special file)

แฟ้มธรรมดา

สำหรับแฟ้มข้อมูลที่อยู่ในยูนิกซ์ จะมีลักษณะที่แตกต่างจากแฟ้มข้อมูลที่อยู่ในระบบปฏิบัติการอื่น คือข้อมูลแต่ละตัวจะอยู่เรียงลำดับกันไปไม่มีลักษณะของระเบียน (record) หรือตัวจบแฟ้มข้อมูล (file terminator) แต่ภายในยูนิกซ์ก็เอื้อให้สามารถเข้าหาข้อมูลทั้งในลักษณะแบบเรียงลำดับ (sequential) กับแบบสุ่ม (random) โดยอาศัยตัวชี้ของแฟ้มข้อมูลช่วยการแทรกข้อมูลหรือการลบข้อมูลที่อยู่กลางๆ แฟ้มข้อมูลนั้นไม่อาจทำได้ซึ่งถ้าหากจำเป็นผู้ใช้ก็ต้องสร้างแฟ้มข้อมูลนั้นขึ้นมาใหม่

แฟ้มข้อมูลเหล่านี้จะมีหมายเลขประจำแฟ้มอยู่เรียกว่าไอ-นัมเบอร์ (i-number)ซึ่งจะเป็นตัวชี้ไปยังรายการในแถวลำดับ ที่เรียกว่าบัพไอ (i-node) ที่เก็บรายละเอียดของแฟ้มข้อมูลนั้น เช่น ประเภทของแฟ้ม ขนาดของแฟ้ม วันที่ที่สร้าง วันที่ที่ปรับปรุงครั้งสุดท้าย

สารบบแฟ้มข้อมูล

สารบบ ก็คือแฟ้มข้อมูลที่เก็บชื่อของแฟ้มข้อมูลต่าง เอาไว้ (ซึ่งอาจเป็นชื่อของสารบบก็ได้) เพื่อใช้ในการแบ่งแยกแฟ้มข้อมูลเหล่านั้นออกเป็นกลุ่มโดยที่สามารถจะแสดงความสัมพันธ์ของกลุ่มเหล่านั้นได้ การที่สารบบ สามารถมี สารบบ ซ้อนอยู่ได้ ทำให้โครงสร้างของแฟ้มข้อมูลต่างๆ อยู่ในลักษณะลำดับขั้น เช่นเดียวกับโครงสร้างแบบต้นไม้ (tree) ซึ่งสารบบที่อยู่บนสุดคือราก (root,/) ภายในสารบบ จะเก็บเพียงไอ-นัมเบอร์ (i-number (2 ไบต์)) กับชื่อของแฟ้มข้อมูล การเรียกใช้แฟ้มข้อมูลจะต้องระบุเส้นทาง (pathname) ซึ่งประกอบด้วยชื่อของสารบบ ที่อยู่ตั้งแต่บนสุดเรียงลำดับลงมาซึ่งแต่ละสารบบ ต้องคั่นด้วยเครื่องหมาย "/"เช่น /dir1/fileเมื่อทำการเรียกใช้แฟ้มใด เคอร์เนลจะหาค่าของไอ-นัมเบอร์ของสารบบแรกจากราก(root)เพื่อเอาไปค้นหาในตารางบัพไอ จากตัวอย่าง เคอร์เนล ทำการค้นหาชื่อdir1 จากสารบบรากซึ่งจะได้ไอ-นัมเบอร์ แล้ว นำไปค้นในตารางบัพไอ เพื่อหาตำแหน่งของ dir1 ในดิสก์ จากนั้นก็ค้นหาชื่อของ จากนั้นก็ค้นหาชื่อ file ในdir1ได้ค่าบัพไอ เพื่อนำไปค้นในตารางบัพไออีกเพื่อหาตำแหน่งของ file ในดิสก์

แฟ้มข้อมูลพิเศษ

แฟ้มข้อมูลนี้จะแตกต่างจากแฟ้มข้อมูลธรรมดาตรงที่ไม่ได้ทำการเก็บข้อมูล แต่แฟ้มข้อมูลประเภทนี้เป็นเพียงชื่อที่แทนอุปกรณ์ต่างๆ ที่ต่อเข้ากับระบบเช่น จานบันทึกข้อมูล แถบบันทึกข้อมูล เทอร์มินัล เครื่องพิมพ์ ซึ่งจะอยู่ในสารบบ /dev เช่น /dev/tty00 ซึ่งหมายถึง พอร์ตเทอร์มินัลหมายเลข 0

แฟ้มข้อมูลพิเศษในยูนิกซ์แบ่งได้ออกเป็น 2 ประเภทคือ อุปกรณ์ประเภทบล็อก (block device) และ อุปกรณ์ประเภทตัวอักษร (character device)

1. อุปกรณ์ประเภทบล็อก

อุปกรณ์ประเภทบล็อก ได้แก่ จานบันทึกข้อมูล เทป การส่งผ่านข้อมูลจากอุปกรณ์ไปยังเคอร์เนลจะทำเป็นบล็อกซึ่งมีขนาดคงที่ (512 ไบท์ต่อ 1 บล็อก) โดยบนอุปกรณ์จะต้องมีบัฟเฟอร์ที่มีขนาดคงที่และในเคอร์เนลก็จำต้องมีกลุ่มของบัฟเฟอร์ใช้เป็นหน่วยความจำแคช (cache) เพื่อเพิ่มความเร็วของรับข้อมูลเข้า/แสดงผล การเข้าถึงข้อมูลทำได้โดยแบบสุ่ม

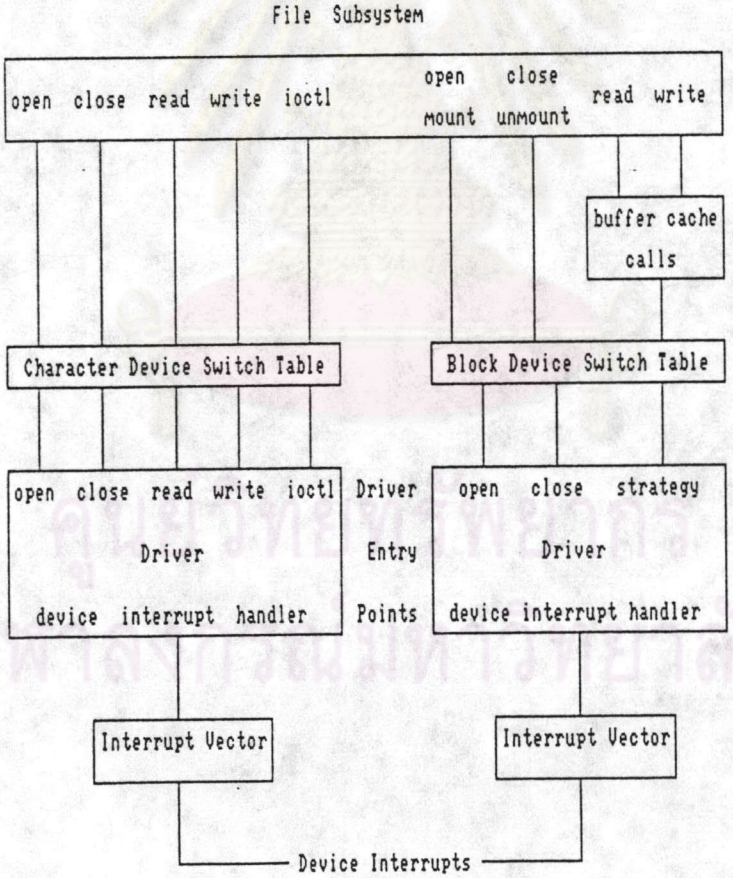
2. อุปกรณ์ประเภทตัวอักษร

อุปกรณ์ประเภทตัวอักษร ได้แก่ เทอร์มินัล เครื่องพิมพ์ ซึ่งอุปกรณ์เหล่านี้จะส่งผ่านข้อมูลแบบสายข้อมูล และไม่อาจเข้าถึงข้อมูลแบบสุ่มได้

ในส่วนของจอควบคุมระบบนี้ จะจัดอยู่ในประเภทของอุปกรณ์ประเภทตัวอักษรแต่จะมีลักษณะพิเศษ คือ สามารถแสดงผลทางด้านกราฟิกได้ด้วย โดยขึ้นอยู่กับชนิดของแผ่นวงจรควบคุมการแสดงผลที่ใช้ว่าสามารถแสดงผลแบบกราฟิกได้ในระดับใด

สำหรับแฟ้มข้อมูลพิเศษจะมีรายการในตารางบัฟไอ ซึ่งจะเก็บค่าหมายเลขของอุปกรณ์(device number) แทนรายการจำนวนข้อมูลในแฟ้มของแฟ้มข้อมูลธรรมดา ซึ่งค่านี้จะถูกใช้โดยเคอร์เนลเป็นดัชนีชี้รายการในตาราง เพื่อหารูทีนของโปรแกรมย่อยควบคุมอุปกรณ์ ตารางที่กล่าวถึงนี้มีอยู่ 2 ตาราง คือ ตารางแสดงอุปกรณ์แบบบล็อก (block

devices switch table) และ ตารางแสดงอุปกรณ์แบบตัวอักษร (character devices switch table) ซึ่งตารางทั้งสองนี้ทำหน้าที่เป็นตัวประสาน (interface) ระหว่างเคอร์เนลกับโปรแกรมย่อยควบคุมอุปกรณ์ นั่นคือเมื่อมีการเรียกใช้ คำสั่งระบบ (system call) เช่น read() หรือ write() จากโปรแกรมของผู้ใช้ เคอร์เนลจะหาค่าหมายเลขของอุปกรณ์ ที่ต้องการอ่านหรือบันทึกจากตารางบัฟไอ แล้วตรวจสอบว่าประเภทของแฟ้มข้อมูลว่าเป็นประเภทใด เพื่อเลือกว่าจะหาตำแหน่งของรูทีนที่ทำหน้าที่ในการอ่านหรือบันทึกข้อมูล จากตารางแสดงอุปกรณ์แบบบล็อก หรือ ตารางแสดงอุปกรณ์แบบตัวอักษร



รูปที่ 3.1 องค์ประกอบของโปรแกรมย่อยควบคุมอุปกรณ์

โปรแกรมย่อยควบคุมอุปกรณ์

อาจกล่าวได้ว่าโปรแกรมย่อยควบคุมอุปกรณ์เป็นกลุ่มของโปรแกรมย่อยและข้อมูลที่อยู่ภายในเคอร์เนล ทำหน้าที่เป็นตัวเชื่อมโยงระหว่างโปรแกรมของผู้ใช้ กับ ตัวอุปกรณ์ โปรแกรมย่อยนี้เขียนด้วยภาษาซี แล้วแปลเป็นโมดูลประสงค์ (object module)เพื่อจะเชื่อมโยงเข้ากับส่วนอื่นของเคอร์เนลด้วยโปรแกรมบรรจุ (loader) นั่นคือถ้าหากต้องการเขียนโปรแกรมย่อยควบคุมอุปกรณ์ขึ้นใหม่ หรือต้องการเปลี่ยนแปลงในตัวโปรแกรมย่อยควบคุมอุปกรณ์เหล่านั้น จะต้องเชื่อมโยงส่วนต่างๆ ใหม่เพื่อสร้างเคอร์เนลใหม่

การเรียกใช้โปรแกรมย่อยควบคุมอุปกรณ์ทำได้โดย เรียกผ่านจุดเข้ามาตรฐาน (standard entry point) ซึ่งจุดเข้าจะแตกต่างกันไปตามประเภทและลักษณะของอุปกรณ์ เช่น เครื่องพิมพ์ ก็ไม่จำเป็นต้องมีฟังก์ชั่นสำหรับการอ่านข้อมูล

จุดเข้าที่พบโดยทั่วไปในโปรแกรมย่อยควบคุมอุปกรณ์

- รูทีนการเริ่มต้นจะถูกเรียกเมื่ออยู่ในระหว่างการเริ่มเปิดระบบ
- รูทีนการเปิดและปิด
- รูทีนการอ่านและเขียน
- รูทีนที่จัดการกับการขัดจังหวะ
- รูทีนที่จัดการประสานจังหวะไอ/โอ หลายทาง (synchronous i/o multiplexing)

โปรแกรมย่อยควบคุมอุปกรณ์ สามารถแบ่งได้ออกตามประเภทของอุปกรณ์ คือ โปรแกรมย่อยควบคุมอุปกรณ์แบบบล็อก และ โปรแกรมย่อยควบคุมอุปกรณ์แบบตัวอักษร ซึ่งต่อไปจะขอกล่าวถึงลักษณะของโปรแกรมย่อยควบคุมจอควบคุมซึ่งเป็นโปรแกรมย่อยควบคุมอุปกรณ์แบบตัวอักษรแบบหนึ่ง

โปรแกรมย่อยควบคุมจอควบคุม

สำหรับจอควบคุมนั้นถือว่าเป็นแฟ้มข้อมูลแบบพิเศษ ซึ่งโปรแกรมของผู้ใช้สามารถอ่านหรือบันทึกได้เช่นเดียวกับแฟ้มข้อมูลอื่นโดยผ่านชื่อของแฟ้มข้อมูล /dev/console หรือผ่านหน่วยรับเข้า/ส่งออกมาตรฐาน หรือ หน่วยแสดงข้อผิดพลาดมาตรฐาน ที่กำหนดให้อยู่แล้วสำหรับแต่ละกระบวนการที่ทำงานบนจอควบคุม

การส่งข้อมูลจากโปรแกรมไปแสดงบนจอภาพ จะถูกกรองโดยเคอร์เนลก่อนที่จะส่งไปที่จอควบคุม ในทำนองเดียวกันการอ่านข้อมูลจากจอควบคุม ข้อมูลเหล่านั้นจะต้องผ่านเคอร์เนลก่อนที่จะส่งให้กับโปรแกรม

ส่วนของเคอร์เนลที่กล่างถึงนี้คือโปรแกรมย่อยควบคุมจอควบคุมซึ่งจะกล่าวถึงหน้าที่ของโปรแกรมย่อยควบคุมจอควบคุมอย่างคร่าวๆ ได้ดังนี้

- กรณีส่งตัวอักขระไปยังจอควบคุม (write)

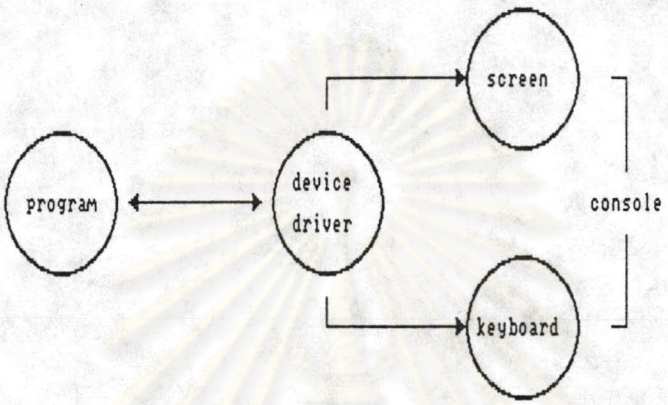
แปลงตัวอักขระเล็กเป็นตัวอักขระใหญ่ ขยายตัวอักขระกั้นวรรคเป็นช่องว่าง แปลงตัวอักขระขึ้นบรรทัดใหม่หรือปัดแคร่เป็นตัวอักขระขึ้นบรรทัดใหม่และปัดแคร่ กรองบิตที่ 8 ของแต่ละตัวอักขระเพื่อใช้เป็นบิตภาวะคู่หรือคี่ (parity bit) ในลักษณะ ภาวะคู่ (odd parity) หรือ ภาวะคี่ (even parity) หรือไม่มี ภาวะคู่หรือคี่ (none parity)

- กรณีรับตัวอักขระจากจอควบคุม (read)

แปลงตัวอักขระขึ้นบรรทัดใหม่และปัดแคร่ เป็นตัวอักขระขึ้นบรรทัดใหม่ ตรวจสอบภาวะคู่หรือคี่ของตัวอักขระแต่ละตัวที่เข้ามา ไม่รับตัวอักขระที่มีภาวะผิด นอกจากนี้แปลงตัวอักขระบางตัวเป็นการกระทำแบบพิเศษ และไม่ส่งให้กับโปรแกรม เช่น ctrl-S ใช้หยุดการแสดงผลบนจอควบคุม ctrl-Q ให้เริ่มแสดงผลต่อ ctrl-C เป็นสัญญานขัดจังหวะเพื่อเลิกการทำงาน ctrl-D ใช้เพื่อบอกว่าจบข้อมูลที่มาจากแป้นอักขระของจอควบคุม

นอกจากนี้โปรแกรมย่อยควบคุมจอควบคุมยังจัดการทางด้านฮาร์ดแวร์ เช่น การแสดงผลบนจอภาพในภาวะกราฟิก การควบคุมภาวะของอุปกรณ์ควบคุมจอภาพ

สิ่งต่างๆ ที่ได้กล่าวนี้สามารถเปลี่ยนแปลงได้โดยผ่านคำสั่ง stty จากเชลล์ หรือคำสั่งระบบ ioctl จากโปรแกรมของผู้ใช้ ซึ่งจะกล่าวถึงต่อไปในภายหลัง

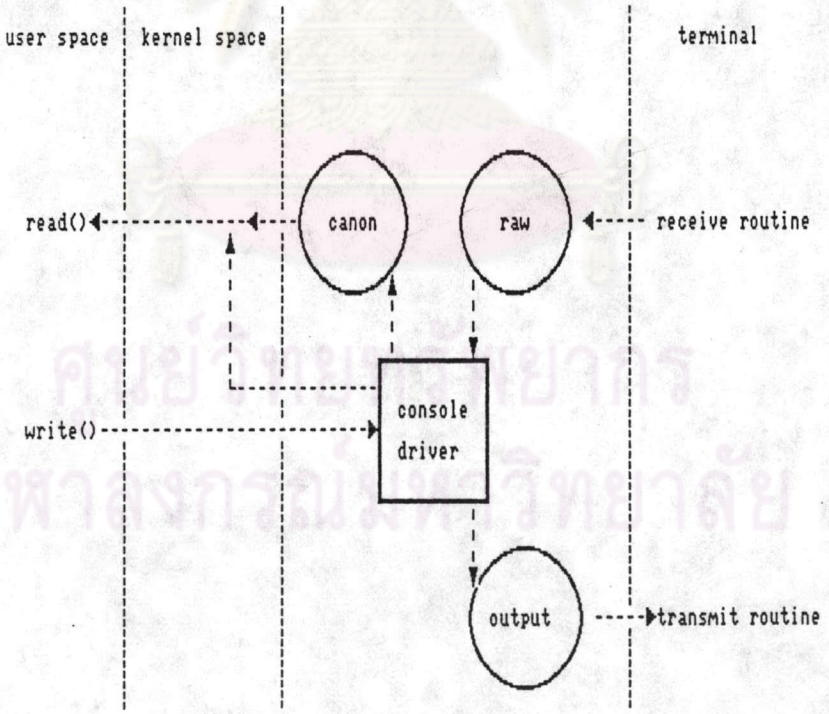


รูปที่ 3.2 ความสัมพันธ์ระหว่างระบบยูนิกซ์กับจอควบคุม

จากรูปที่3.2 เมื่อโปรแกรมของผู้ใช้เรียกฟังก์ชั่นเขียน write() โปรแกรมย่อยควบคุมจอควบคุม จะนำเอาข้อมูลเหล่านั้นไปแสดงบนจอภาพพร้อมทั้งตีความตัวอักขระพิเศษ (เช่นก้นวรรค ขึ้นบรรทัดใหม่) ในทำนองเดียวกันเมื่อโปรแกรมเรียกฟังก์ชั่นอ่าน read() โปรแกรมนี้จะอ่านข้อมูลจากแป้นพิมพ์พร้อมทั้งแสดงข้อมูลเหล่านั้นออกทางจอภาพ แต่ข้อมูลจะยังไม่ส่งให้กับโปรแกรมของผู้ใช้จนกว่าผู้ใช้กดแป้นขึ้นบรรทัด (return key)

การทำงานระหว่างโปรแกรมย่อยควบคุมอุปกรณ์กับจอควบคุมเป็นแบบสองทาง(full duplex)ดังนั้นการแสดงผลออกทางจอภาพจะเป็นหน้าที่ของโปรแกรมย่อยควบคุมจอควบคุม มิได้เป็นหน้าที่ของจอควบคุม เมื่อมีการป้อนข้อมูลเข้ามาทีละอักขระ ข้อมูลก็จะถูกส่งไปให้โปรแกรมย่อยควบคุมจอควบคุม ซึ่งจะทำสำเนาข้อมูลเหล่านั้นแล้วแสดงออกทางจอภาพ แต่ข้อมูลจะยังไม่ถูกส่งไปให้กับโปรแกรมของผู้ใช้ จนกว่าโปรแกรมของผู้ใช้พร้อมที่จะอ่านข้อมูล ซึ่งถ้าหากว่าในขณะที่โปรแกรมผู้ใช้สั่งเขียนออกบนจอภาพ เป็นเวลาเดียวกันกับที่ผู้ใช้ป้อนข้อมูลเข้ามาก็จะแสดงออกทางจอภาพปนกับข้อมูลที่สั่งเขียนจากโปรแกรม และเมื่อ

โปรแกรมสั่งอ่านข้อมูลข้อมูลที่ถูกป้อนเข้ามาเหล่านั้นจะถูกส่งไปให้กับโปรแกรม การที่โปรแกรมย่อยควบคุม จอควบคุมสามารถทำงานในลักษณะนี้ได้ ต้องอาศัยบัฟเฟอร์ที่มีโครงสร้างเป็นรายการโยง (linked list) ซึ่งมีอยู่ด้วยกัน 3 ชุด คือ บัฟเฟอร์รอว์ บัฟเฟอร์คานอน และบัฟเฟอร์ส่งออก แต่การทำงานของโปรแกรมย่อยควบคุมจอควบคุม ยังขึ้นอยู่กับภาวะของจอควบคุมซึ่งมีอยู่ด้วยกัน 2 ภาวะ คือ ภาวะคาโนนิคัล และภาวะรอว์ โดยจะมีผลต่อการแสดงผลบนจอภาพ และข้อมูลที่จะส่งให้กับโปรแกรมของผู้ใช้ เช่น ต้องการอ่านข้อมูลจากแป้นพิมพ์ขณะที่จอควบคุมอยู่ในภาวะคาโนนิคัล ถ้าหากผู้ใช้ต้องการพิมพ์ตัวอักขระ d a t e แต่ป้อนเป็น d s t e แทน ผู้ใช้ต้องป้อนตัวอักขระถอยหลัง (backspace) 3 ตัว และ ตัว a t e ใหม่ ซึ่งในที่สุดโปรแกรมของผู้ใช้จะได้ตัวอักขระ 4 ตัว คือ date แต่ถ้าหากจอควบคุมอยู่ในภาวะรอว์ โปรแกรมของผู้ใช้จะได้ ตัวอักขระทั้งหมด 11 ตัว ลักษณะการส่งผ่านข้อมูลจากจอควบคุมไปยังโปรแกรมของผู้ใช้ แสดงได้ดังรูปที่ 3.3



รูปที่ 3.3 การไหลของข้อมูลในโปรแกรมย่อยควบคุมจอควบคุม

จอควบคุมในภาวะคาโนนิคัล

เมื่อโปรแกรมของผู้ใช้เรียกฟังก์ชั่นในการเขียน write() ข้อมูลที่อยู่ในพื้นที่ว่างของผู้ใช้ (user space) จะถูกส่งไปให้โปรแกรมย่อยควบคุมจอควบคุม เพื่อที่จะส่งและแปลงตัวอักขระพิเศษ (เช่น กั้นวรรคจะถูกแปลงเป็นช่องว่างหลายตัวติดกัน) ไปเก็บไว้ในบัฟเฟอร์ส่งออก จนกว่าข้อมูลที่อยู่ในบัฟเฟอร์ส่งออกมีมากจนถึงจุดที่เรียกว่าจุดระดับน้ำสูง (high water mark) จึงเรียกโปรแกรมย่อยควบคุมที่ทำหน้าที่เอาผลออกแสดงทางจอภาพให้ทำงาน ซึ่งในขณะนี้โปรแกรมย่อยควบคุมจอควบคุมก็จะถูกปลุกให้ขึ้นมาทำงานต่อเป็นอย่างนี้เรื่อยไปจนกว่าข้อมูลที่ส่งมาจากโปรแกรมของผู้ใช้จะหมด

ส่วนการทำงานของโปรแกรมย่อยควบคุมจอควบคุม เมื่อเรียกฟังก์ชั่นในการอ่าน read() ค่อนข้างจะซับซ้อนมากกว่าขั้นตอนการเขียน write() โดยหลังจากที่ถูกเรียกฟังก์ชั่นในการอ่าน read() จะตรวจสอบในบัฟเฟอร์รอว์ว่ามีข้อมูลหรือไม่ ถ้าหากไม่มีโปรแกรมย่อยนี้ก็จะหลับจนกว่าจะได้รับการแจ้งจาก ส่วนควบคุมสัญญานการขัดจังหวะของเทอร์มินัล (terminal interrupt handler) ส่วนควบคุมสัญญานการขัดจังหวะของเทอร์มินัล ถูกปลุกด้วยสัญญาณการขัดจังหวะจากตัวควบคุมเทอร์มินัล (terminal contoller) ซึ่งจะเก็บข้อมูลในบัฟเฟอร์รอว์ไปเรื่อยๆ พร้อมทั้งส่งข้อมูลไปยังบัฟเฟอร์ส่งออกเพื่อแสดงตัวอักขระที่ถูกป้อนเข้ามาออกทางจอภาพ โดยจะแปลงตัวอักขระพิเศษต่างๆ เช่น อักขระลบ (erase character) ถูกแปลงเป็นตัวอักขระ 3 ตัว คือ ถอยหลัง เว้นวรรค ถอยหลัง (backspace space backspace) ให้อยู่ในรูปแบบที่ผู้ใช้เข้าใจได้ แต่ถ้าหากอักขระพิเศษที่ป้อนเข้ามาคือ ตัวปัดแคร่ (carriage return) โปรแกรมย่อยนี้จะหยุดการดึงข้อมูลจากจอควบคุมใส่ลงในบัฟเฟอร์ แต่จะดึงข้อมูลจากบัฟเฟอร์รอว์ไปใส่ในบัฟเฟอร์คานอนแทน โดยจะแปลงตัวอักขระพิเศษ เช่น อักขระลบก็จะลบข้อมูลที่อยู่ในบัฟเฟอร์คานอน 1 ตัว หรือ ตัวปัดแคร่ จะถูกแปลงเป็นตัวอักขระขึ้นบรรทัดใหม่) เพื่อส่งให้กับโปรแกรมของผู้ใช้หลังจากที่ดึงข้อมูลจากบัฟเฟอร์รอว์มาตรวจสอบจนหมด

จอควบคุมในภาวะรอว์

สำหรับในภาวะรอว์นั้นข้อมูลจะถูกส่งไปให้โปรแกรมของผู้ใช้เพียง 1 ตัวอักขระเมื่อทำการเรียกฟังก์ชั่นในการอ่าน read() 1 ครั้ง ตัวอักขระทุกตัวจะถูกเก็บลงในบัฟเฟอร์รอว์ แล้วส่งให้กับโปรแกรมของผู้ใช้ พร้อมทั้งแสดงออกทางจอภาพโดยไม่มีการแปลงตัวอักขระพิเศษดังเช่น ในภาวะคาโนนิคัล ทำให้ตัวปัดแคร่ถูกมองเป็นข้อมูลให้กับโปรแกรมของผู้ใช้แทนการมองเป็นตัวจบบรรทัดของการป้อนข้อมูล ลักษณะเช่นนี้จะทำให้เหมาะกับการเขียนโปรแกรมที่ต้องการอ่านข้อมูล โดยไม่ต้องรอเคาะตัวปัดแคร่ เช่น โปรแกรมบรรณาธิการ vi สามารถใช้คำสั่งเป็นตัวอักขระใดๆ แทนตัวอักขระควบคุม (control character) ได้