รูปภาพประกอบด้วย สัญลักษณ์, ยอด, เครื่องหมาย, มงกุฎ

คำอธิบายที่สร้างโดยอัตโนมัติ

**ใบงานที่ 2**

**เรื่อง Process management(Fork)**

**เสนอ**

**อาจารย์ปิยพล ยืนยงสถาวร**

**จัดทำโดย**

**นายอธิศ สุนทโรดม**

**รหัส 65543206086-2**

**ใบงานนี้เป็นส่วนหนึ่งของวิชาระบบปฏิบัติการ (ENGCE125)**

**หลักสูตรวิศกรมมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์**

**มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา เชียงใหม่**

**ประจำภาคเรียนที่ 2 ปีการศึกษา 2566**

**การเขียนใบงานการทดลอง**

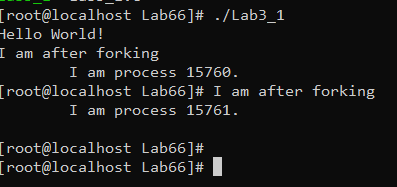
1. เขียนอธิบายขั้นตอนการทำงานโปรแกรมพร้อมแคปเจอร์รูปภาพขั้นตอนการทำงานสรุปผลการทำงานในแต่ละหัวข้อของการทดลอง

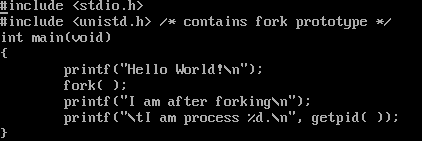
2. ทำเป็นเอกสารใบงานบันทึกงานเป็นไฟล์ Wordและปริ้นส่ง

3. จากการทดลองมีข้อแตกต่างกันอย่างไรบ้างจงอธิบาย

4. สรุปผลการทดลอง

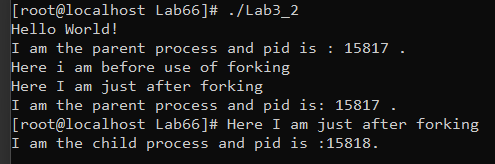
**ตัวอย่างที่ 1**

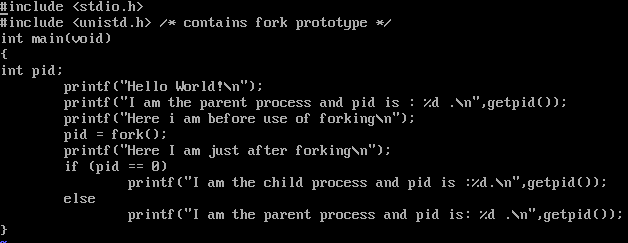




โค้ดในภาพเป็นโปรแกรมที่แสดงข้อความ "Hello World!" โดยใช้ฟังก์ชัน fork() ของระบบปฏิบัติการฟังก์ชัน fork() จะสร้างกระบวนการใหม่ขึ้นมาจากกระบวนการปัจจุบัน Processใหม่นี้จะเรียกว่า " Process ลูก" และกระบวนการปัจจุบันจะเรียกว่า " Process พ่อ" โดย Process ทั้ง 2 จะมีคุณสมบัติเหมือนกันแต่จะแตกต่างกันที่หมายเลข id โดยแสดงออกมากโดย getpid()

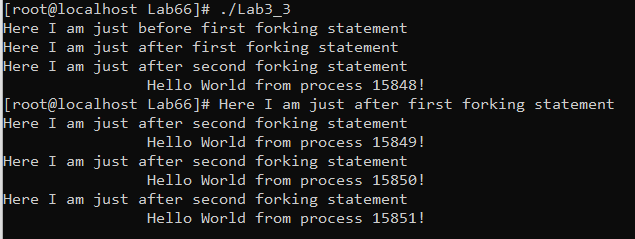
**ตัวอย่างที่ 2**

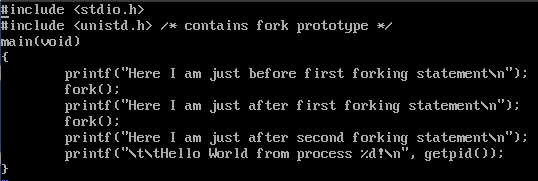




โค้ดในภาพนี้แสดงถึงการสร้างforkของ parent procees และ child process โดยเริ่มต้นจะทำการพิมprocess id ของ parent จากนั้นทำการบอกผู้ใช้ว่ากำลังจะทำการสร้าง process ของ child ต่อจากนี้จากนั้นทำการสร้าง fork() โดยเก็บค่าไว้ใน pid จากนั้นเช็คค่าของ pid เมื่อมีค่าเท่ากับ 0 ให้แสดง process id ของ child

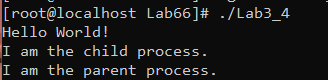
**ตัวอย่างที่ 3**

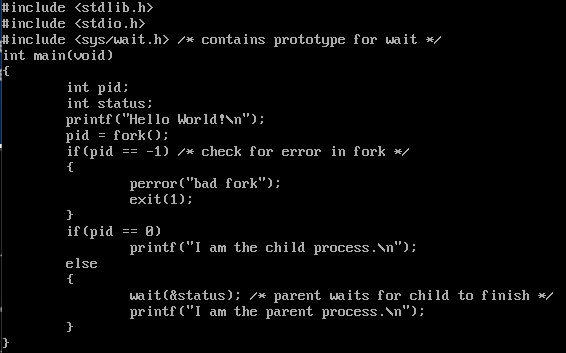




โค้ดในภาพนี้แสดงถึงการสร้าง fork หลาย process จาก process หลักโดยจะทำงานตาม fork ที่ถูกสร้างขึ้นโดยทั้งหมดมีการสร้าง 2 process โดยและมีการบอกผู้ใช้ถึงการสร้าง process ลูก โดยเมื่อมีการรันHello World จะมีการบอก id ของ process นั้นๆเพื่อระบุว่าและแยกแยะความแตกต่างระหว่าง process

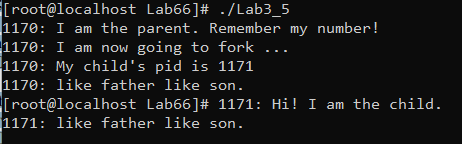
**ตัวอย่างที่ 4**

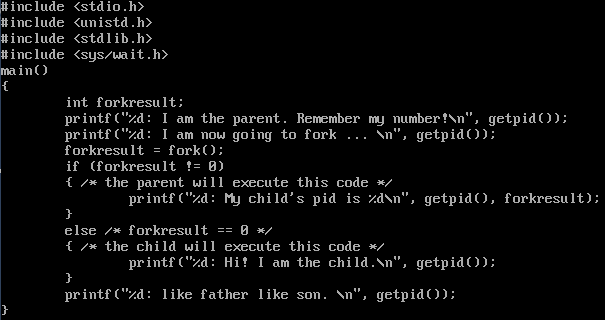




โค้ดในภาพนี้แสดงถึงการตอบโต้ระหว่าง process พ่อ และ ลูก โดยจะมีการเก็บค่าของ pid และ status ไว้จากนั้นทำการสร้าง process ลูกและทำการตรวจสอบโดย if (pid == -1) ตรวจสอบว่าfork()เกิดข้อผิดพลาดหรือไม่ หากเกิดข้อผิดพลาด ให้พิมพ์ข้อความแสดงข้อผิดพลาด "bad fork" และ exit(1) เพื่อยุติโปรแกรมหากว่าไม่เกิดข้อผิดพลาดจะทำการเช็กว่าเป็น process ลูกหรือไม่โดยถ้า pid == 0 คือเมื่อส่งค่า 0 จะแสดงว่าเป็น process ลูก แต่ถ้าไม่จะเรียกใช้waitฟังก์ชันเพื่อรอให้กระบวนการลูกเสร็จสิ้นและ&statusอาร์กิวเมนต์ได้รับสถานะการออกของกระบวนการลูกจากนั้นให้แสดงว่าเป็น parent process

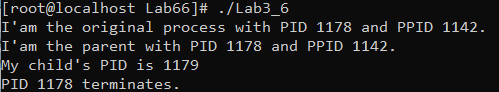
**ตัวอย่างที่ 5**

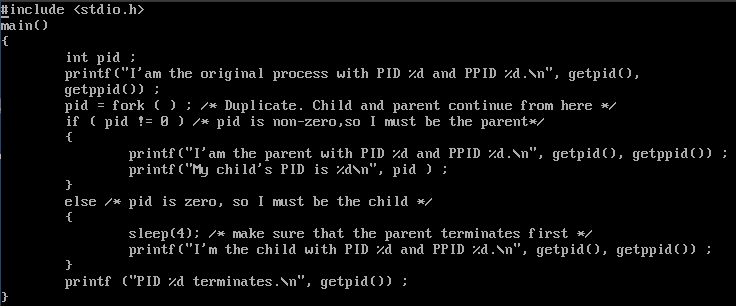




โค้ดในภาพนี้แสดงถึงการแยกกันทำงานของ process ลูกโดยจะเก็บค่าไว้ใน forkresult และเมื่อforkresult != 0 จะเป็นการทำงานของ process พ่อ โดยจะแสดง process id ของ Child แต่เมื่อไม่ตรงเงื่อนไขนี้หรือ forkresult = 0 จะทำงานอีกเงื่อนไขโดยพิมพ์ข้อความที่ระบุถึงการดำเนินการของ process ลูก

**ตัวอย่างที่ 6**





โค้ดในภาพนี้แสดงถึง id ของ process เป็นขั้นๆ โดยเมื่อเริ่มต้นจะประกาศตัวแปรจำนวนเต็มpidเพื่อจัดเก็บ ID กระบวนการของกระบวนการลูก และทำการแสดง getpid(): ID กระบวนการปัจจุบันและgetppid():ID กระบวนการหลักของกระบวนการปัจจุบันจากนั้นจะทำการ fork และทำตามเงื่อนไขโดยเมื่อ pid != 0 จะแสดงว่าเป็น parent process และแสดง id ของ parent และ child และถ้าเป็นอีกเงื่อนไขจะแสดงว่าเป็น child process จะรอเป็นเวลา 4 วินาทีเพื่อให้แน่ใจว่ากระบวนการหลักเสร็จสิ้นก่อนและพิมพ์ข้อความ " I'm the child with...”

**ตัวอย่างที่ 7**

A computer screen shot of a black screen

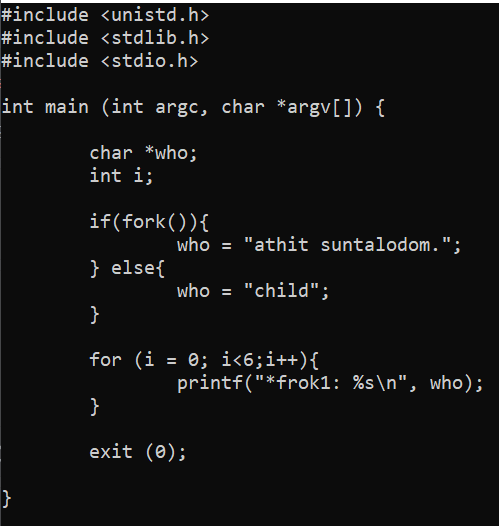
Description automatically generated

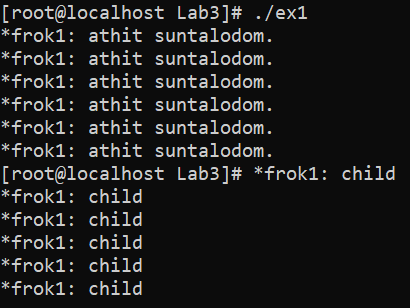
A computer screen with white text

Description automatically generated

โค้ดในภาพนี้แสดงถึงการทำงานของ process และการทำงานแบบ background process โดยทำการสร้าง fork ออกมาและจากนั้นทำการวนลูปการทำงาน 1 อย่างเพื่อทดสอบการทำงานของ background process จากนั้นทำการ run โดย a.out เพื่อดูการทำงานของ process โดย 1268 เป็น process ที่ทำงานแบบ background process โดยทุก 10 วิจะพิมชื่อออกมาและสามารถทำการสั่งการทำงานอื่นๆได้โดยในที่นี้จะทำการสั่ง kill 1268 เพื่อจะให้หยุดการทำงาน

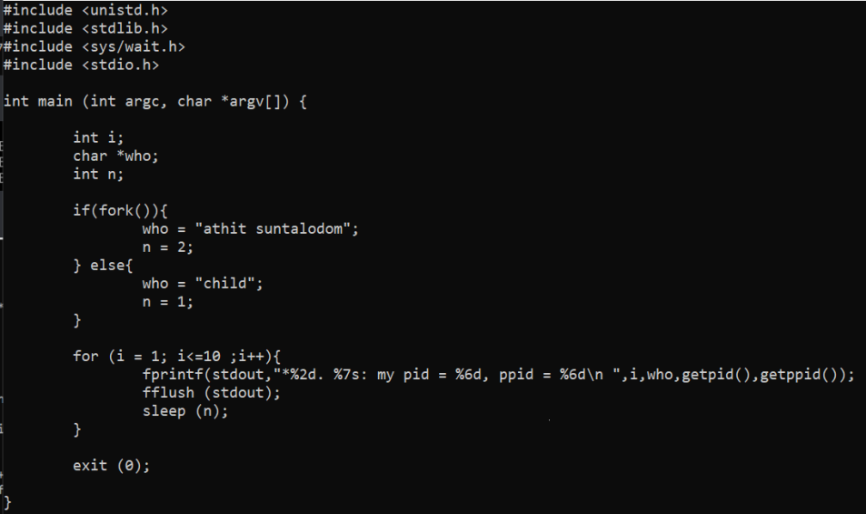
**แบบฝึกหัดที่ 1**

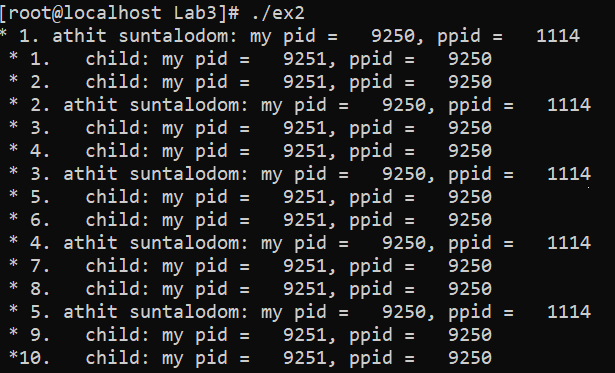




โค้ดในภาพนี้แสดงถึงการทำงานของ process โดยจะระบุเป็นชื่อกับ childโดยเมื่อเริ่มต้นจะประกาศตัวแปรเก็บค่าของ int และ char เพื่อใช้ในการแสดงผล เก็บข้อมูล และเช็คเงื่อนไขโดยเมื่อทำการ fork ครั้งแรกจะการรับ who เป็นชื่อและนำมาแสดง 6 รอบโดยจะเป็นการทำงานของ parent process ก่อนและทำการ fork จากนั้นทำการแสดงชื่อของ child เมื่อเราทำการ fork ไปแล้ว 6 รอบในการทำงานยี้เป็นส่วนของ child process

**แบบฝึกหัดที่ 2**



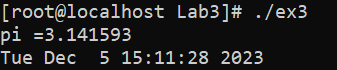


โค้ดในภาพนี้แสดงถึง pid ของ process ทั้งหมดโดยเริ่มจากการกำหนดให้ชื่อเป็น parent process และ child process จากนั้นกำหนดการทำงานโดยจะทำการวนลูป print ชื่อ และ parent id และ ppid โดย child จะ print ออกมาเป็นจำนวน 10 ครั้งทุกๆ 1วิ และชื่อและเป็น 10 ครั้ง ทุกๆ 2 วิ

**แบบฝึกหัดที่ 3**

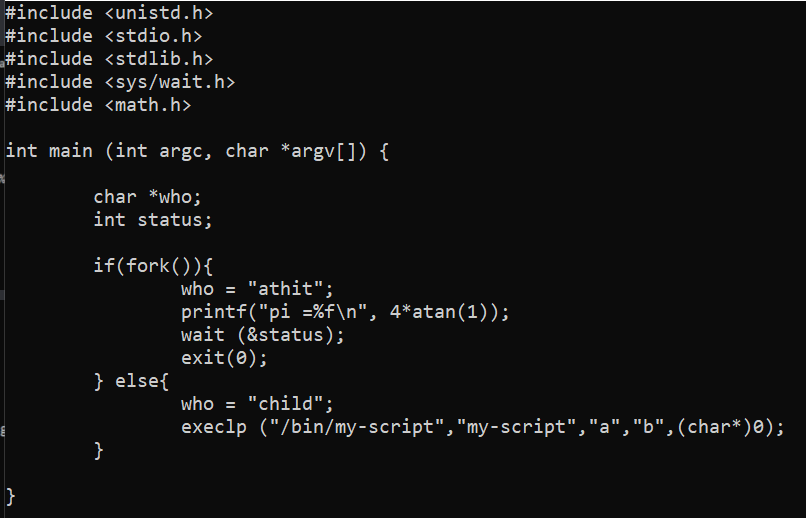
A computer screen shot of white text

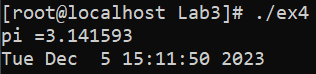
Description automatically generated



โค้ดในภาพนี้แสดงถึงการทำงานของ child process โดนจะทำการประการ who และทำการ fork จากนั้นให้แสดงค่าของ pi และทำการ wait เพื่อเก็บค่าและรอ child หยุดการทำงานจากนั้นเมื่อทำงานอีกครั้งจะทำงานโดย child จะเข้าไปค้นหาตาม path และ print ค่าออกมาเป็น วัน/เวลา

**แบบฝึกหัดที่ 4**





โค้ดในภาพนี้แสดงถึงการทำงานของ child process โดยจะประกาศ who เพื่อให้การทำงานของ Processแยกกันและทำการ fork จากนั้นจะแสดง pic และทำการ wait เพื่อเก็บค่าและรอ child หยุดการทำงานจากนั้นจะทำการเรียกใช้งาน script เพื่อแสดงวัน/เวลา

**จากการทดลอง มีข้อแตกต่างกันอย่างไร**

โดยการใช้ fork() จะทำในส่วน parent process ส่วนของ child process จะทำงานหลังจากที่สร้างขึ้น parent process ก็จะทำต่อโดยการทำงานและทำตามคำสั่งเหมือนกันโดยอยู่ที่เราจะกำหนดแต่ id ของ process จะไม่เหมือนกัน

**สรุปผลการทดลอง**

ฟังก์ชัน fork() ในภาษา C เป็นฟังก์ชันพื้นฐานสำหรับการสร้างกระบวนการใหม่ กระบวนการใหม่นี้เรียกว่า (child process) และกระบวนการเดิมเรียกว่า (parent process)

การทำงานของฟังก์ชัน fork() มีดังนี้

ระบบปฏิบัติการจะสร้างกระบวนการใหม่ขึ้นมา โดยกระบวนการใหม่นี้จะมีลักษณะเหมือนกับ parent process ทุกประการ ยกเว้นค่าตัวแปร pid ซึ่งจะแตกต่างกันฟังก์ชัน

fork() จะคืนค่ากลับไปยัง parent process ในกรณีที่การสร้างกระบวนการใหม่สำเร็จ ฟังก์ชัน fork() จะคืนค่าเป็นค่ากระบวนการลูก ในกรณีที่การสร้างกระบวนการใหม่ล้มเหลว ฟังก์ชัน fork() จะคืนค่าเป็นค่า -1 ใน parent process เราสามารถตรวจสอบค่าที่คืนกลับมาจากฟังก์ชัน fork() เพื่อดูว่าการสร้างกระบวนการใหม่สำเร็จหรือไม่ ในกรณีที่การสร้างกระบวนการใหม่สำเร็จ เราสามารถเรียกใช้ฟังก์ชัน wait() เพื่อรอให้กระบวนการลูกทำงานเสร็จก่อนจึงจะดำเนินการต่อได้สำหรับกระบวนการลูก

กระบวนการลูกจะเริ่มต้นการทำงานที่จุดที่ฟังก์ชัน fork() คืนค่ากลับไปยัง parent process กระบวนการลูกสามารถทำงานแยกจาก parent process ได้อย่างอิสระ ตัวอย่างเช่น กระบวนการลูกสามารถทำงานต่าง ๆ ร่วมกัน ทำงานต่างเวลา หรือทำงานต่าง ๆ กันกับ parent process ก็ได้