

作業系統概論 hw7

學號: 408410113 姓名: 王 X 彥

執行環境:

Operating System: Kubuntu 20.04

KDE Plasma Version: 5.18.5

KDE Frameworks Version: 5.68.0

Qt Version: 5.12.8

Kernel Version: 5.11.0-37-generic

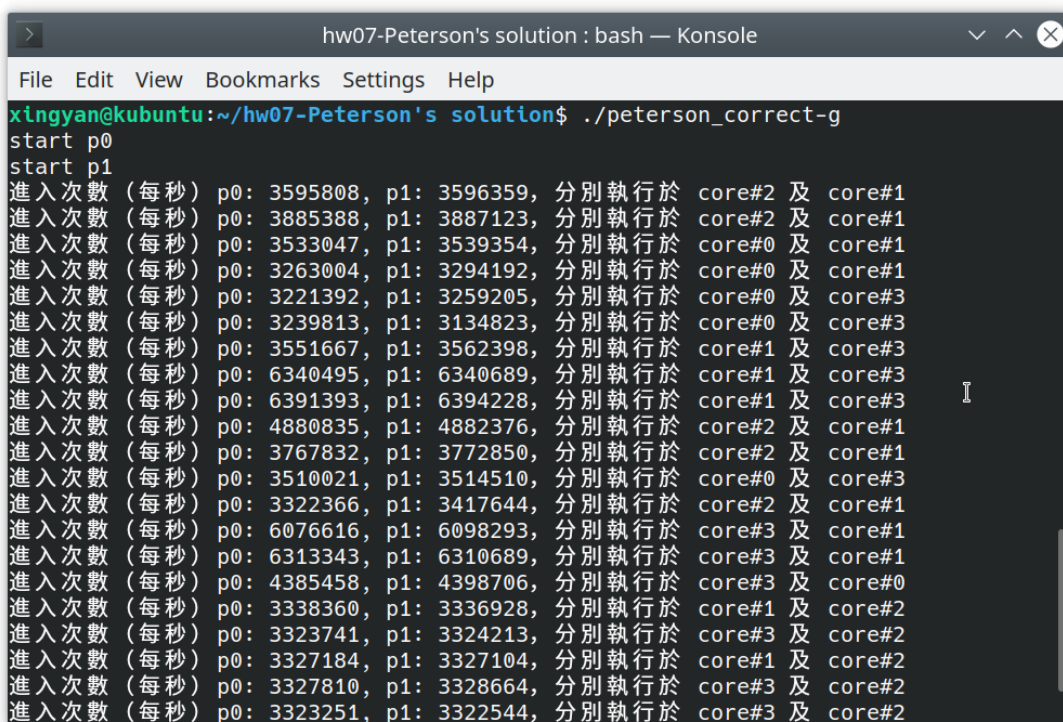
OS Type: 64-bit

Processors: 4 x Intel® Core™ i7-6500U CPU @ 2.50GHz

Memory: 11.6 GiB of RAM

1. 執行make, 之後會產生四個執行檔案。請問你的執行結果為何? 請附上畫面截圖

(1) ./peterson_correct-g



```
hw07-Peterson's solution : bash — Konsole
File Edit View Bookmarks Settings Help
xingyan@kubuntu:~/hw07-Peterson's solution$ ./peterson_correct-g
start p0
start p1
進入次數 (每秒) p0: 3595808, p1: 3596359, 分別執行於 core#2 及 core#1
進入次數 (每秒) p0: 3885388, p1: 3887123, 分別執行於 core#2 及 core#1
進入次數 (每秒) p0: 3533047, p1: 3539354, 分別執行於 core#0 及 core#1
進入次數 (每秒) p0: 3263004, p1: 3294192, 分別執行於 core#0 及 core#1
進入次數 (每秒) p0: 3221392, p1: 3259205, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3239813, p1: 3134823, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3551667, p1: 3562398, 分別執行於 core#1 及 core#3
進入次數 (每秒) p0: 6340495, p1: 6340689, 分別執行於 core#1 及 core#3
進入次數 (每秒) p0: 6391393, p1: 6394228, 分別執行於 core#1 及 core#3
進入次數 (每秒) p0: 4880835, p1: 4882376, 分別執行於 core#2 及 core#1
進入次數 (每秒) p0: 3767832, p1: 3772850, 分別執行於 core#2 及 core#1
進入次數 (每秒) p0: 3510021, p1: 3514510, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3322366, p1: 3417644, 分別執行於 core#2 及 core#1
進入次數 (每秒) p0: 6076616, p1: 6098293, 分別執行於 core#3 及 core#1
進入次數 (每秒) p0: 6313343, p1: 6310689, 分別執行於 core#3 及 core#1
進入次數 (每秒) p0: 4385458, p1: 4398706, 分別執行於 core#3 及 core#0
進入次數 (每秒) p0: 3338360, p1: 3336928, 分別執行於 core#1 及 core#2
進入次數 (每秒) p0: 3323741, p1: 3324213, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 3327184, p1: 3327104, 分別執行於 core#1 及 core#2
進入次數 (每秒) p0: 3327810, p1: 3328664, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 3323251, p1: 3322544, 分別執行於 core#3 及 core#2
```

(2) ./peterson_correct-03

```
hw07-Peterson's solution : bash — Konsole
File Edit View Bookmarks Settings Help
xingyan@kubuntu:~/hw07-Peterson's solution$ ./peterson_correct-03
start p0
start p1
進入次數 (每秒) p0: 3272053, p1: 3271933, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3376008, p1: 3376918, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3405522, p1: 3407583, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3404261, p1: 3409835, 分別執行於 core#0 及 core#1
進入次數 (每秒) p0: 3393896, p1: 3393610, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3582288, p1: 3588492, 分別執行於 core#0 及 core#1
進入次數 (每秒) p0: 3384711, p1: 3443925, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3305334, p1: 3306561, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3390761, p1: 3387062, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3415240, p1: 3419944, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3284187, p1: 3285403, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3245294, p1: 3336137, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3435140, p1: 3435764, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3299309, p1: 3292676, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3404456, p1: 3401789, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3342876, p1: 3385302, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3265090, p1: 3284749, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3113827, p1: 3263733, 分別執行於 core#0 及 core#3
進入次數 (每秒) p0: 3334359, p1: 3336465, 分別執行於 core#2 及 core#3
進入次數 (每秒) p0: 3377804, p1: 3382121, 分別執行於 core#2 及 core#1
進入次數 (每秒) p0: 3297698, p1: 3288643, 分別執行於 core#2 及 core#3
```

(3) ./peterson_trival-g

```
hw07-Peterson's solution : bash — Konsole
File Edit View Bookmarks Settings Help
xingyan@kubuntu:~/hw07-Peterson's solution$ ./peterson_trival-g
p0: start
p1: start
p0及p1都在critical section
進入次數 (每秒) p0: 5734463, p1: 5733927, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5794844, p1: 5794849, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5798271, p1: 5798272, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5797329, p1: 5797339, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5793811, p1: 5793817, 分別執行於 core#3 及 core#0
進入次數 (每秒) p0: 11587648, p1: 5793788, 分別執行於 core#3 及 core#0
進入次數 (每秒) p0: 5794455, p1: 5794530, 分別執行於 core#3 及 core#0
進入次數 (每秒) p0: 11585501, p1: 5790988, 分別執行於 core#3 及 core#0
進入次數 (每秒) p0: 5794569, p1: 5794562, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5798912, p1: 5798907, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5792859, p1: 5792868, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 11587962, p1: 5795000, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 17378991, p1: 5790898, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5790417, p1: 5790437, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 11570760, p1: 5780291, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 5792115, p1: 5792119, 分別執行於 core#3 及 core#0
進入次數 (每秒) p0: 5763113, p1: 5763134, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 11555927, p1: 5792724, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 17331180, p1: 5775345, 分別執行於 core#3 及 core#2
進入次數 (每秒) p0: 23123321, p1: 5792097, 分別執行於 core#3 及 core#2
```

(4) ./peterson_trival-O3

```
hw07-Peterson's solution : bash — Konsole
File Edit View Bookmarks Settings Help
xingyan@kubuntu:~/hw07-Peterson's solution$ ./peterson_trival-O3
p0: start
p1: start
進入次數 (每秒) p0: 1120, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
進入次數 (每秒) p0: 0, p1: 0, 分別執行於 core#1 及 core#0
^C
```

2. 「確實的」解釋「為什麼」peterson_trival-O3 的執行結果是錯的

因為從反組譯的結果可以看到經過 O3 的優化後，turn 這個變數的判斷被優化掉了，變成 while 只需判斷 flag，而 turn 的判斷永遠都為 true，因此造成錯誤。

```
hw07-Peterson's solution : gdb — Konsole
File Edit View Bookmarks Settings Help
Dump of assembler code for function p0:
0x0000000000001360 <+0>:   endbr64
0x0000000000001364 <+4>:   push    %rbx
0x0000000000001365 <+5>:   lea     0xd13(%rip),%rdi      # 0x207f
0x000000000000136c <+12>:  lea     0xcd(%rip),%rbx      # 0x2060
0x0000000000001373 <+19>:  callq   0x10e0 <puts@plt>
0x0000000000001378 <+24>:  cmpl    $0x1,0x2cbd(%rip)    # 0x403c <flag1>
0x000000000000137f <+31>:  movl    $0x1,0x2caf(%rip)    # 0x4038 <flag0>
0x0000000000001389 <+41>:  movl    $0x1,0x2cad(%rip)    # 0x4040 <turn>
0x0000000000001393 <+51>:  jne     0x13e4 <p0+132>
0x0000000000001395 <+53>:  nopl    (%rax)
0x0000000000001398 <+56>:  jmp     0x1398 <p0+56>
0x000000000000139a <+58>:  nopw    0x0(%rax,%rax,1)
0x00000000000013a0 <+64>:  mov     0x2c79(%rip),%rcx    # 0x4020 <stderr@GLIBC_2.2.5>
0x00000000000013a7 <+71>:  mov     $0x1e,%edx
0x00000000000013ac <+76>:  mov     $0x1,%esi
0x00000000000013b1 <+81>:  mov     %rbx,%rdi
0x00000000000013b4 <+84>:  callq   0x1150 <fwrite@plt>
0x00000000000013b9 <+89>:  addl    $0x1,0x2c6c(%rip)    # 0x402c <p0_in_cs>
0x00000000000013c0 <+96>:  subl    $0x1,0x2c6d(%rip)    # 0x4034 <in_cs>
0x00000000000013c7 <+103>: cmpl    $0x1,0x2c6e(%rip)    # 0x403c <flag1>
0x00000000000013ce <+110>: movl    $0x1,0x2c60(%rip)    # 0x4038 <flag0>
0x00000000000013d8 <+120>: movl    $0x1,0x2c5e(%rip)    # 0x4040 <turn>
0x00000000000013e2 <+130>: je      0x1398 <p0+56>
0x00000000000013e4 <+132>: callq   0x1140 <sched_getcpu@plt>
0x00000000000013e9 <+137>: mov     0x2c45(%rip),%edx    # 0x4034 <in_cs>
0x00000000000013ef <+143>: mov     %eax,0x2c4f(%rip)    # 0x4044 <cpu_p0>
0x00000000000013f5 <+149>: lea     0x1(%rdx),%eax
0x00000000000013f8 <+152>: mov     %eax,0x2c36(%rip)    # 0x4034 <in_cs>
0x00000000000013fe <+158>: cmp     $0x2,%eax
0x0000000000001401 <+161>: je      0x13a0 <p0+64>
0x0000000000001403 <+163>: addl    $0x1,0x2c22(%rip)    # 0x402c <p0_in_cs>
0x000000000000140a <+170>: cmpl    $0x1,0x2c2b(%rip)    # 0x403c <flag1>
0x0000000000001411 <+177>: mov     %edx,0x2c1d(%rip)    # 0x4034 <in_cs>
0x0000000000001417 <+183>: movl    $0x1,0x2c17(%rip)    # 0x4038 <flag0>
0x0000000000001421 <+193>: movl    $0x1,0x2c15(%rip)    # 0x4040 <turn>
0x000000000000142b <+203>: jne     0x13e4 <p0+132>
0x000000000000142d <+205>: jmpq    0x1398 <p0+56>
--Type <RET> for more, q to quit, c to continue without paging--
```

3. 請問在你的電腦上「peterson_trival-g」的速度比「peterson_correct-O3」快或者是賣？上述二個程式的正確與否？

在我的電腦上peterson_trival-g進入CS的次數較多因此推測速度較快，但是執行時會發生P0和P1同時在CS，因此結果不正確。

4. 請「確實的」解釋「題三」，某個程式比另一個程式快或者慢的理由。

(1) 反組譯peterson_trival-g中的p0

```
hw07-Peterson's solution : gdb — Konsole
File Edit View Bookmarks Settings Help
Dump of assembler code for function p0:
0x00000000000012b7 <+0>:    endbr64
0x00000000000012bb <+4>:    push    %rbp
0x00000000000012bc <+5>:    mov     %rsp,%rbp
0x00000000000012bf <+8>:    lea     0xd9a(%rip),%rdi        # 0x2060
0x00000000000012c6 <+15>:   callq   0x10e0 <puts@plt>
0x00000000000012cb <+20>:   movl    $0x1,0x2d5f(%rip)      # 0x4034 <flag0>
0x00000000000012d5 <+30>:   movl    $0x1,0x2d4d(%rip)      # 0x402c <turn>
0x00000000000012df <+40>:   nop
0x00000000000012e0 <+41>:   mov     0x2d4a(%rip),%eax      # 0x4030 <flag1>
0x00000000000012e6 <+47>:   cmp     $0x1,%eax
0x00000000000012e9 <+50>:   jne     0x12f6 <p0+63>
0x00000000000012eb <+52>:   mov     0x2d3b(%rip),%eax      # 0x402c <turn>
0x00000000000012f1 <+58>:   cmp     $0x1,%eax
0x00000000000012f4 <+61>:   je      0x12e0 <p0+41>
0x00000000000012f6 <+63>:   callq   0x1140 <sched_getcpu@plt>
0x00000000000012fb <+68>:   mov     %eax,0x2d43(%rip)      # 0x4044 <cpu_p0>
0x0000000000001301 <+74>:   mov     0x2d31(%rip),%eax      # 0x4038 <in_cs>
0x0000000000001307 <+80>:   add     $0x1,%eax
0x000000000000130a <+83>:   mov     %eax,0x2d28(%rip)      # 0x4038 <in_cs>
0x0000000000001310 <+89>:   mov     0x2d22(%rip),%eax      # 0x4038 <in_cs>
0x0000000000001316 <+95>:   cmp     $0x2,%eax
0x0000000000001319 <+98>:   jne     0x133b <p0+132>
0x000000000000131b <+100>:  mov     0x2cfe(%rip),%rax      # 0x4020 <stderr@GLIBC_2.2.5>
0x0000000000001322 <+107>:  mov     %rax,%rcx
0x0000000000001325 <+110>:  mov     $0x1e,%edx
0x000000000000132a <+115>:  mov     $0x1,%esi
0x000000000000132f <+120>:  lea     0xd3a(%rip),%rdi      # 0x2070
0x0000000000001336 <+127>:  callq   0x1150 <fwrite@plt>
0x000000000000133b <+132>:  mov     0x2cff(%rip),%eax      # 0x4040 <p0_in_cs>
0x0000000000001341 <+138>:  add     $0x1,%eax
0x0000000000001344 <+141>:  mov     %eax,0x2cf6(%rip)      # 0x4040 <p0_in_cs>
0x000000000000134a <+147>:  mov     0x2ce8(%rip),%eax      # 0x4038 <in_cs>
0x0000000000001350 <+153>:  sub     $0x1,%eax
0x0000000000001353 <+156>:  mov     %eax,0x2cdf(%rip)      # 0x4038 <in_cs>
0x0000000000001359 <+162>:  movl    $0x0,0x2cd1(%rip)      # 0x4034 <flag0>
0x0000000000001363 <+172>:  jmpq    0x12cb <p0+20>
End of assembler dump.
(gdb) █
```


(2) 反組譯peterson_correct-O3中的p0

```
hw07-Peterson's solution : gdb — Konsole
File Edit View Bookmarks Settings Help
Dump of assembler code for function p0:
0x0000000000001370 <+0>:    endbr64
0x0000000000001374 <+4>:    push   %rax
0x0000000000001375 <+5>:    pop    %rax
0x0000000000001376 <+6>:    lea    0xd02(%rip),%rdi    # 0x207f
0x000000000000137d <+13>:   sub    $0x8,%rsp
0x0000000000001381 <+17>:   callq  0x10e0 <puts@plt>
0x0000000000001386 <+22>:   nopw   %cs:0x0(%rax,%rax,1)
0x0000000000001390 <+32>:   movl   $0x1,0x2ca6(%rip)    # 0x4040 <flag>
0x000000000000139a <+42>:   mfence
0x000000000000139d <+45>:   mfence
0x00000000000013a0 <+48>:   movl   $0x1,0x2c9e(%rip)    # 0x4048 <turn>
0x00000000000013aa <+58>:   mfence
0x00000000000013ad <+61>:   jmp     0x13bb <p0+75>
0x00000000000013af <+63>:   nop
0x00000000000013b0 <+64>:   mov     0x2c92(%rip),%eax    # 0x4048 <turn>
0x00000000000013b6 <+70>:   cmp     $0x1,%eax
0x00000000000013b9 <+73>:   jne     0x13c5 <p0+85>
0x00000000000013bb <+75>:   mov     0x2c83(%rip),%eax    # 0x4044 <flag+4>
0x00000000000013c1 <+81>:   test    %eax,%eax
0x00000000000013c3 <+83>:   jne     0x13b0 <p0+64>
0x00000000000013c5 <+85>:   callq   0x1140 <sched_getcpu@plt>
0x00000000000013ca <+90>:   mov     %eax,0x2c84(%rip)    # 0x4054 <cpu_p0>
0x00000000000013d0 <+96>:   mov     0x2c62(%rip),%eax    # 0x4038 <in_cs>
0x00000000000013d6 <+102>:  lea     0x1(%rax),%edx
0x00000000000013d9 <+105>:  mov     %edx,0x2c59(%rip)    # 0x4038 <in_cs>
0x00000000000013df <+111>:  cmp     $0x2,%edx
0x00000000000013e2 <+114>:  jne     0x140a <p0+154>
0x00000000000013e4 <+116>:  mov     0x2c35(%rip),%rcx    # 0x4020 <stderr@GLIBC_2.2.5>
0x00000000000013eb <+123>:  mov     $0x1e,%edx
0x00000000000013f0 <+128>:  mov     $0x1,%esi
0x00000000000013f5 <+133>:  lea     0xc64(%rip),%rdi    # 0x2060
0x00000000000013fc <+140>:  callq   0x1150 <fwrite@plt>
0x0000000000001401 <+145>:  mov     0x2c31(%rip),%eax    # 0x4038 <in_cs>
0x0000000000001407 <+151>:  sub     $0x1,%eax
0x000000000000140a <+154>:  addl    $0x1,0x2c1f(%rip)    # 0x4030 <p0_in_cs>
0x0000000000001411 <+161>:  mov     %eax,0x2c21(%rip)    # 0x4038 <in_cs>
0x0000000000001417 <+167>:  movl    $0x0,0x2c1f(%rip)    # 0x4040 <flag>
--Type <RET> for more, q to quit, c to continue without paging--
```

(3) 在(2)的結果中可以看到為了使程式的執行順序正確，而為使用了 mfence，這樣可以避免 turn

被 O3 優化掉，以達到正確的結果，但也因為這樣必須多花一些步驟，導致餘時間上會比(1)還要慢。