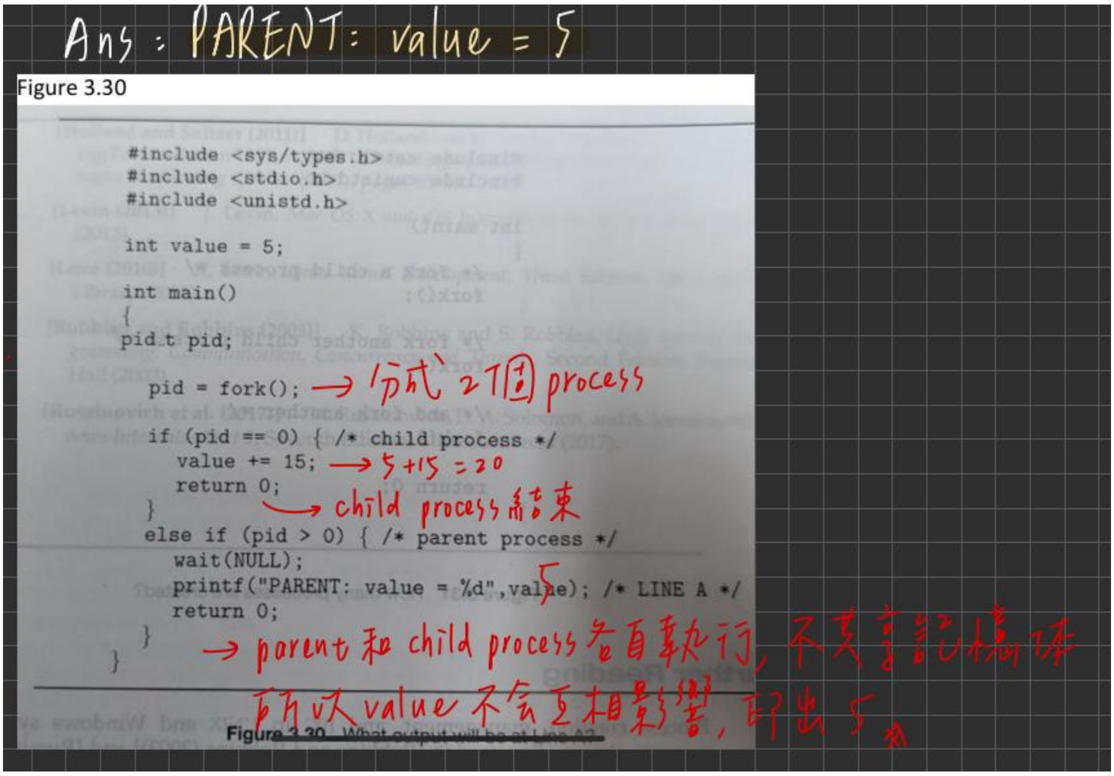
112213080 資管三 蔡秉凱

手寫

3.1



Ans：PARENT: value = 5

Figure 3.30

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main()
{
pid_t pid;

  pid = fork();

  if (pid == 0) { /* child process */
    value += 15;
    return 0;
  }
  else if (pid > 0) { /* parent process */
    wait(NULL);
    printf("PARENT: value = %d",value); /* LINE A */
    return 0;
  }
}
```

pid = fork(); → 分成2個process

value += 15; → 5+15=20

→ child process結束

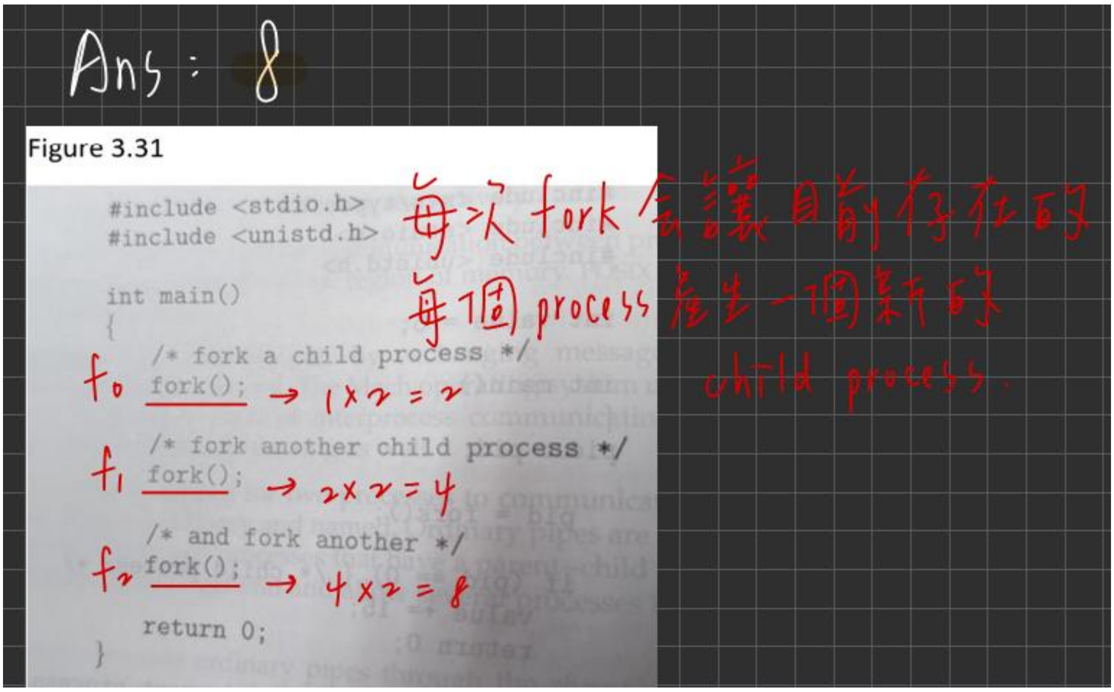
5

→ parent和child process各自執行，不共享記憶体

所以value不会互相影響，印出5。

Figure 3.30 What output will be at Line A?

3.2



Ans：8

Figure 3.31

```
#include <stdio.h>
#include <unistd.h>

int main()
{
  /* fork a child process */
  fork();

  /* fork another child process */
  fork();

  /* and fork another */
  fork();

  return 0;
}
```

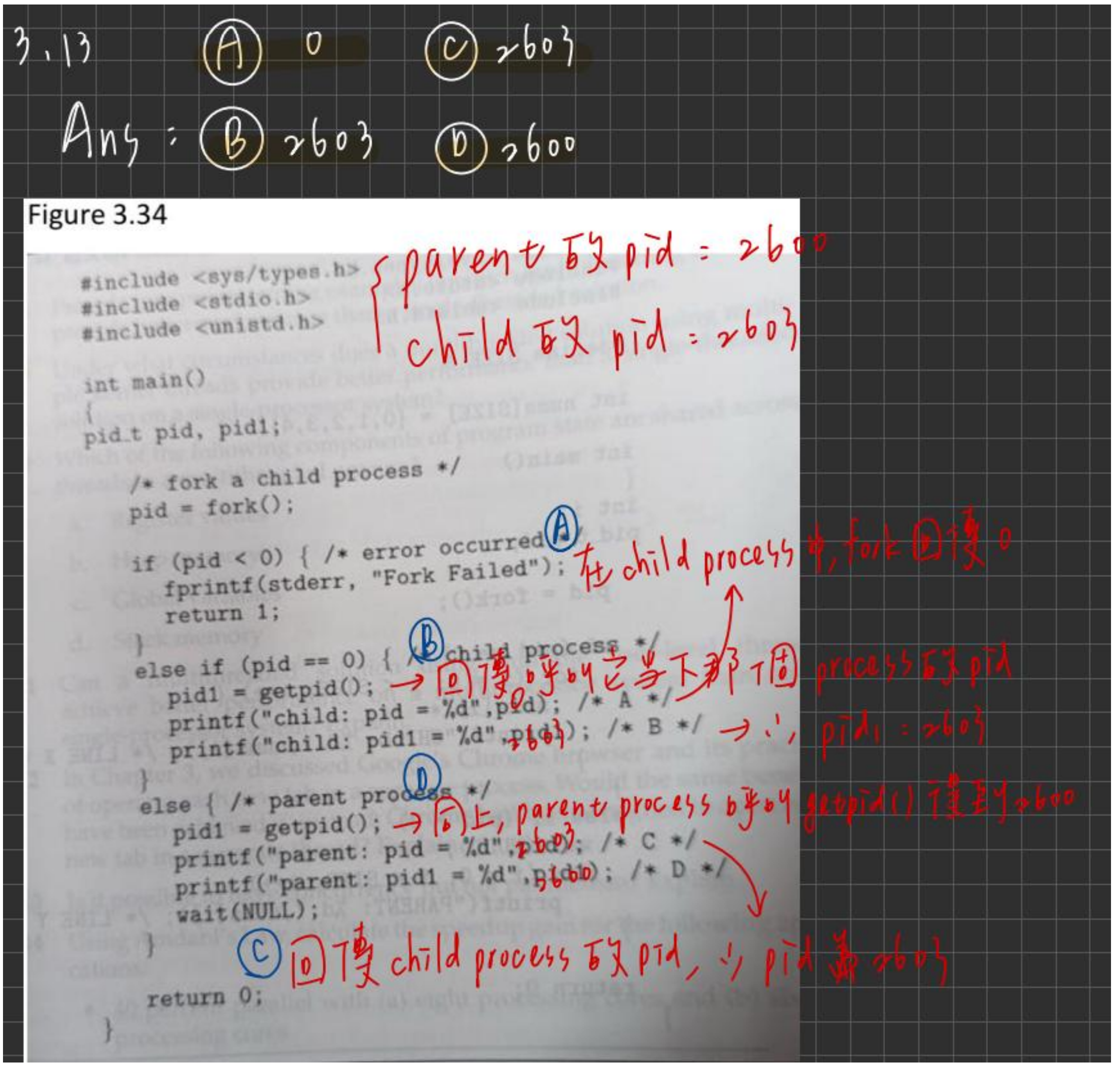
f0 fork(); → 1×2=2

f1 fork(); → 2×2=4

f2 fork(); → 4×2=8

每次fork会讓目前存在的每個process產生一個新的child process。

3.13



3.13 Ⓐ 0 Ⓒ 2603

Ans : Ⓑ 2603 Ⓓ 2600

Figure 3.34

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid, pid1;

   /* fork a child process */
   pid = fork();

   if (pid < 0) { /* error occurred */
     fprintf(stderr, "Fork Failed");
     return 1;
   }
   else if (pid == 0) { /* child process */
     pid1 = getpid();
     printf("child: pid = %d",pid); /* A */
     printf("child: pid1 = %d",pid1); /* B */
   }
   else { /* parent process */
     pid1 = getpid();
     printf("parent: pid = %d",pid); /* C */
     printf("parent: pid1 = %d",pid1); /* D */
     wait(NULL);
   }

   return 0;
}
```

parent 的 pid = 2600

child 的 pid = 2603

Ⓐ 在 child process 中, fork 回傳 0

Ⓑ → 回傳呼叫它當下那個 process 的 pid → 所以, pid1 = 2603

0

2603

Ⓓ → 同上, parent process 呼叫 getpid() 得到 2600

2603

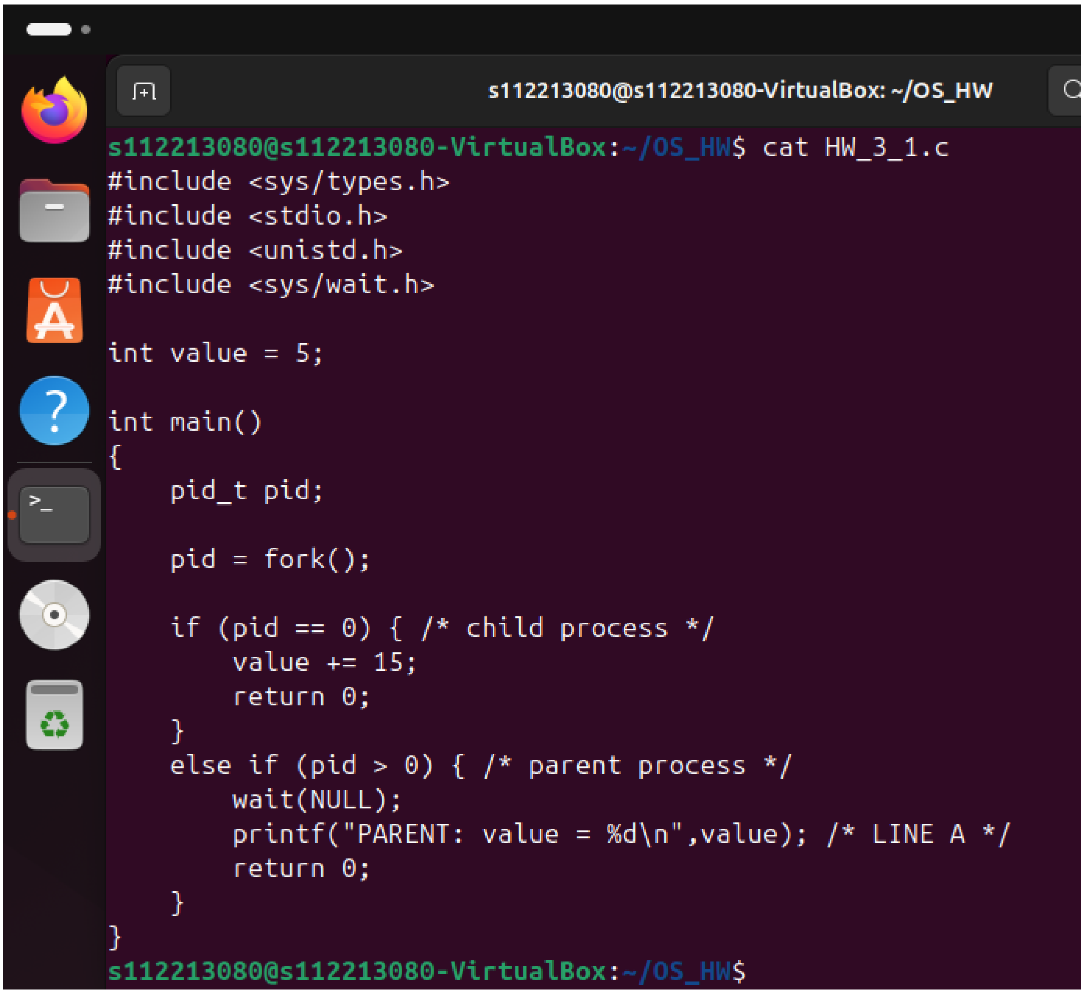
2600

Ⓒ 回傳 child process 的 pid, 所以 pid 為 2603

程式實作

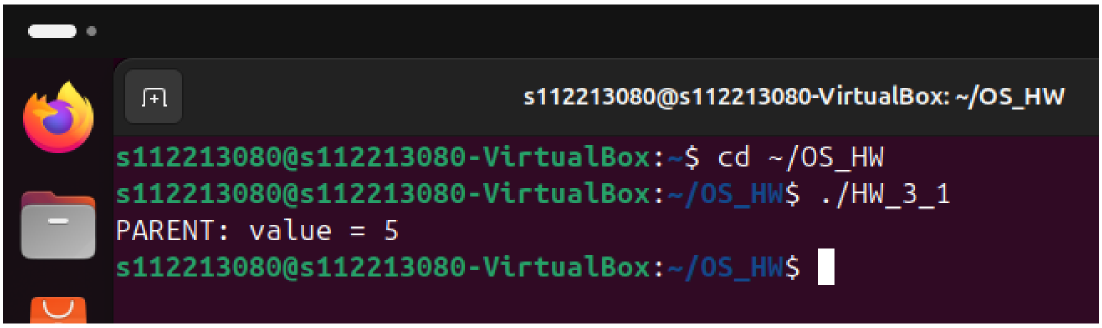
3.1

原始程式：



```
s112213080@s112213080-VirtualBox:~/OS_HW$ cat HW_3_1.c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int value = 5;

int main()
{
    pid_t pid;

    pid = fork();

    if (pid == 0) { /* child process */
        value += 15;
        return 0;
    }
    else if (pid > 0) { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d\n",value); /* LINE A */
        return 0;
    }
}
s112213080@s112213080-VirtualBox:~/OS_HW$
```

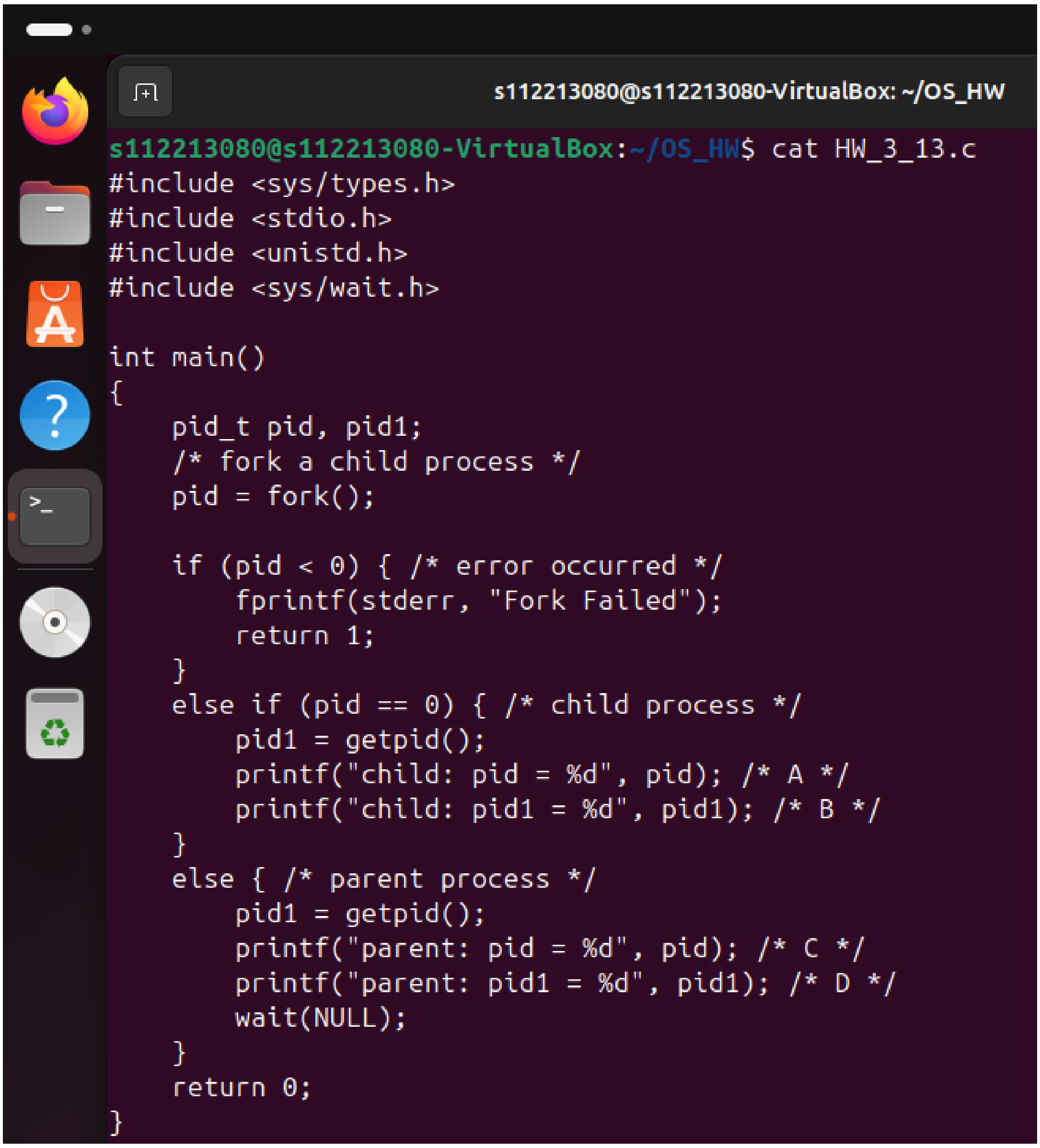
執行結果：



```
s112213080@s112213080-VirtualBox:~$ cd ~/OS_HW
s112213080@s112213080-VirtualBox:~/OS_HW$ ./HW_3_1
PARENT: value = 5
s112213080@s112213080-VirtualBox:~/OS_HW$
```

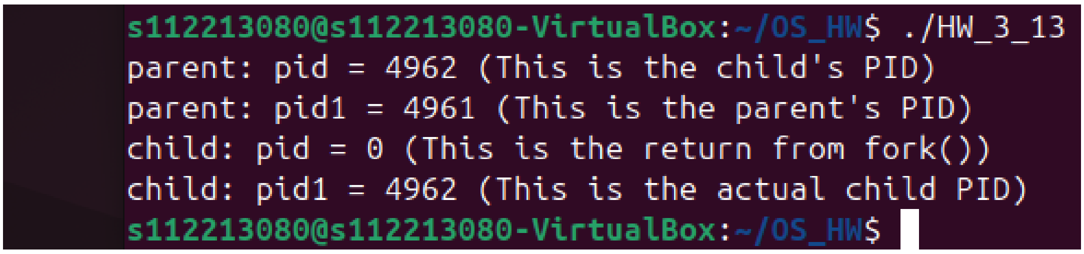
3.13

原始程式：



```
s112213080@s112213080-VirtualBox:~/OS_HW$ cat HW_3_13.c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
    pid_t pid, pid1;
    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d", pid); /* A */
        printf("child: pid1 = %d", pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d", pid); /* C */
        printf("parent: pid1 = %d", pid1); /* D */
        wait(NULL);
    }
    return 0;
}
```

執行結果：



```
s112213080@s112213080-VirtualBox:~/OS_HW$ ./HW_3_13
parent: pid = 4962 (This is the child's PID)
parent: pid1 = 4961 (This is the parent's PID)
child: pid = 0 (This is the return from fork())
child: pid1 = 4962 (This is the actual child PID)
s112213080@s112213080-VirtualBox:~/OS_HW$
```