

PARENT **vs** CHILD

using

`wait` or `waitpid`

Sanjaya Kumar Jena

ITER, Bhubanewar

Text Book(s)



Kay A. Robbins, & Steve Robbins

UnixTM Systems Programming

Communications, concurrency, and Treads

Pearson Education

Reference Book(s)



Brain W. Kernighan, & Rob Pike

The Unix Programming Environment

PHI

Guess the Output-1

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid,waitreturn; /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
    }
    waitreturn=wait(NULL);
    if (childpid != waitreturn) {
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
        printf("Return value of wait=%d\n",waitreturn);
    }
    return 0;
}
```

Guess the Output-2

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid,waitreturn; /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
    }
    waitreturn=wait(NULL);
    if (childpid == waitreturn) {
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
        printf("Return value of wait=%d\n",waitreturn);
    }
    return 0;
}
```

Identify the Parent Code & the Child Code

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid,waitreturn; /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
    }
    waitreturn=wait(NULL);
    if (childpid == waitreturn) {
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
        printf("Return value of wait=%d\n",waitreturn);
    }
    return 0;
}
```

Identification of Parent and Child

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid,waitreturn; /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {                          /* Child Code */
        printf("Process ID=%ld\n", (long)getpid());
    }
    waitreturn=wait(NULL);
    if (childpid == waitreturn) /* Parent Code */
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
        printf("Return value of wait=%d\n",waitreturn);
    }
    return 0;
}
```

Placing wait

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid;                               /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
    }
    if (childpid == wait(NULL))
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
    }
    return 0;
}
```

Find the code part for child and parent

Guess the Output-3

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid;                             /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
    }
    if (childpid != wait(NULL))
        printf("Return value of fork=%ld\n", (long)childpid);
    printf("Process ID=%ld\n", (long)getpid());
}
return 0;
}
```

Find the code part for child and parent.

Guess the Output-3

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid;                             /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
    }
    if (childpid != wait(NULL))
        printf("Return value of fork=%ld\n", (long)childpid);
    printf("Process ID=%ld\n", (long)getpid());
}
return 0;
}
```

Find the code part for child and parent.

Does the parent display any output?

Guess the Output-4

```
int main() {                                /* PID of child 3425 */
    pid_t childpid;                          /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
        return 0;
    }
    if (childpid != wait(NULL))
        printf("Return value of fork=%ld\n", (long)childpid);
    printf("Process ID=%ld\n", (long)getpid());
}
return 0;
}
```

Find the code part for child and parent. Does the parent display any output?



Guess the Output-5

```
int main() {                                /* PID of child 3425 */
    pid_t childpid;                          /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
        return 0;
    }
    if (childpid == wait(NULL))
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
    }
    return 0;
}
```

Find the code part for child and parent. Does the parent display any output?



Guess the Output-6

```
int main() {                                /* PID of child 3425 */
    pid_t childpid;                          /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
        return 0;
    }
    if (childpid != wait(NULL)) {
        printf("Parent failed to wait due to signal/err:\n");
        return 1;
    }
    return 0;
}
```

Find the code part for child and parent. Does the parent display any output?

Putting All Cases Together

```
int main () {
    pid_t childpid;
    childpid = fork();
    if (childpid == -1) {
        fprintf(stderr, "Failed to fork\n");
        return 1;
    }
    if (childpid == 0)
        printf("I am child %ld\n", (long)getpid());
    else if (wait(NULL) != childpid)
        printf("A signal must have interrupted the wait!\n");
    else
        printf("I am parent %ld with child %ld\n", (long)
            getpid(), (long)childpid);
    return 0; }
```

Multiple call to wait

```
int main() {                                     /* PID of child 3425 */
    pid_t childpid;                             /* PID of parent 3424 */
    childpid = fork();
    if(childpid == 0) {
        printf("Process ID=%ld\n", (long)getpid());
        return 0;
    }
    if (childpid == wait(NULL))
        printf("Return value of fork=%ld\n", (long)childpid);
        printf("Process ID=%ld\n", (long)getpid());
        printf("Again return value of wait=%d\n", wait(NULL));
    }
    return 0;
}
```