Subject: Lecture on Operating System Lecture Exercise 7.5
From: IKENOYA Katsutoshi <j05002@ie.u-ryukyu.ac.jp>
Date: Tue, 13 Feb 2007 21:19:42 +0900
To: Shinji KONO <kono@ie.u-ryukyu.ac.jp>

学籍番号：055702B

・修正点

中でwaitしないように修正
子プロセスがゾンビにならないよう修正

問題7.5

server.plを man perlipc などを参考に、fork するように書き換えて見よ。
fork する方が適しているサービスにはどんなものがあるかを考えて実装してみよ。
(C で書く場合は、課題で行なう stream.c の getstream を使うのが良い)
アクセスカウンタを fork する server で作る場合には、lock 等が必要になる。
lock を使う場合はWWWの演習で出て来る。また、アクセスカウンタ用のサーバを
作ることにより、実現することもできる。lock と専用サーバの利点と欠点、
向いているサービスなどについて考察せよ。

・ソースの抜粋

ソースの場所：/net/home/cvs/y05/j05002/OS/Lecture7/mondai7-5/server_fork3.pl
/net/home/cvs/y05/j05002/OS/Lecture7/mondai7-5/client.pl

```
sub spawn;  # 先行宣言
sub logmsg { print "@_ at ", scalar localtime, "¥n" }
```

省略

```
my $paddr;
my $waitedpid=0;
my $pid;

sub REAPER{
    $waitedpid = wait;
    $SIG{CHLD} = ¥&REAPER;  # loathe sysv
    logmsg "reaped $waitedpid" . ($? ? " with exit $?" : '');
}

$SIG{CHLD} = ¥&REAPER;

for ($waitedpid = 0 ; ( $paddr = accept(Client,Server) ) || $waitedpid ;
       $waitedpid = 0 , close Client) {
    next if $waitedpid and not $paddr;

    my($port,$iaddr) = sockaddr_in($paddr);
    my $name = gethostbyaddr($iaddr,AF_INET);

    logmsg "connection from $name [",inet_ntoa($iaddr), "]";

    spawn sub {
         print "Hello there, $name, it's now ",scalar localtime, "¥n¥n";
    };
}

sub spawn {
    my $coderef = shift;
```

```
    unless (@_ == 0 && $coderef && ref($coderef) eq 'CODE') {
        confess "usage: spawn CODEREF";
    }

    my $pid;
    if (!defined($pid = fork)) {
        logmsg "cannot fork: $!";
        return;
    } elsif ($pid) {
        logmsg "begat $pid";
        return;
    }


    open(STDIN,  "<&Client")   || die "can't dup client to stdin";
    open(STDOUT, ">&Client")   || die "can't dup client to stdout";
        ## open(STDERR, ">&STDOUT") || die "can't dup stdout to stderr";
    exit &$coderef();
}
```

・実行結果

```
[j05002@mondai7-5]% perl server_fork3.pl
server started on port 2345 at Sun Feb 11 18:26:36 2007
connection from localhost [ 127.0.0.1 ] at Sun Feb 11 18:26:41 2007
begat 553 at Sun Feb 11 18:26:41 2007
reaped 553 with exit 256 at Sun Feb 11 18:26:41 2007
connection from localhost [ 127.0.0.1 ] at Sun Feb 11 18:26:44 2007
begat 555 at Sun Feb 11 18:26:44 2007
reaped 555 with exit 256 at Sun Feb 11 18:26:44 2007
connection from localhost [ 127.0.0.1 ] at Sun Feb 11 18:26:45 2007
begat 557 at Sun Feb 11 18:26:45 2007
reaped 557 with exit 256 at Sun Feb 11 18:26:45 2007
```

クライアントが接続するたびに、PIDやポート番号
が変わっていることから、forkが正しく動作している
ことが分かる。

このようにforkするサーバープログラムは、複数のクライアントが
同時にサーバにアクセスするようなサービスに向いている。
例えば、HPのアクセスカウンタである。
この場合、アクセスカウンタをサーバー側でforkして作成することになる。
しかし、複数の人が同時に同じファイルにアクセスする可能性がある。
そのため、ファイルのアクセス数などの整合性をとるために、lock等で
ファイル書き込み時に排他制御を行う必要がある。