Ejercicio 2

Tema 2

Ejercicio 2

```
int *pids;
void usage () {
     char  b[128];
     sprintf (b, "./Mtarea procesosnivel1(cuantos)
procesosnivel2 (0=no/1=si) \n");
    write (1,b,strlen(b));
    exit (0);
}
void realizatarea (int i) {
     // código sin llamadas a sistema
}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
              it = 0;
              while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
         ret=fork();
         if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
         pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
          for ( i = 0 ; i < procesos; i++) {
               if ( pids[i] == ret ) {
                    sprintf(buff,
                   "acaba el proceso num %d con pid %d \n" ,i,ret);
                    write(1,buff,strlen(buff));
               }
          }
    }
}
```

1) Dibuja la jerarquía de procesos que se genera al ejecutar:
(asigna identificadores a los procesos para poder referirte después a ellos):

```
>   ./Mtarea 5 0
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
         for ( i = 0 ; i < procesos; i++) {
             if ( pids[i] == ret ) {
                 sprintf(buff,
                 "acaba el proceso num %d con pid %d \n" ,i,ret);
                 write(1,buff,strlen(buff));
             }
         }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```


```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

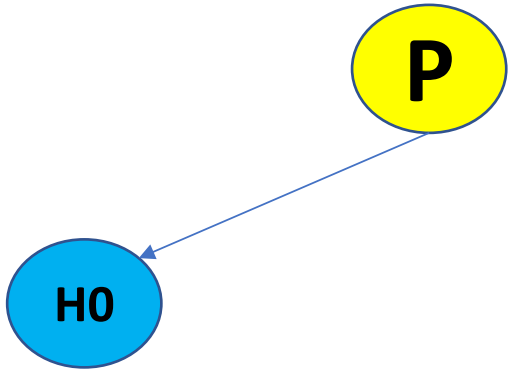
```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```


```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

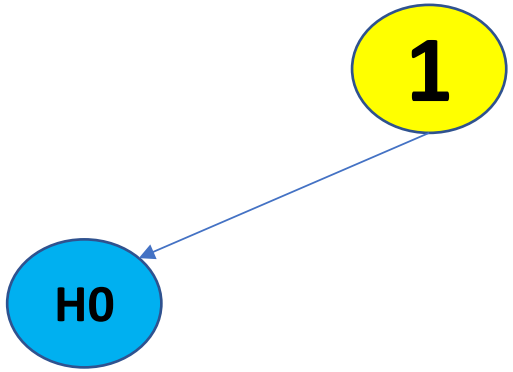


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
          for ( i = 0 ; i < procesos; i++) {
               if ( pids[i] == ret ) {
                    sprintf(buff,
                   "acaba el proceso num %d con pid %d \n" ,i,ret);
                    write(1,buff,strlen(buff));
               }
          }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

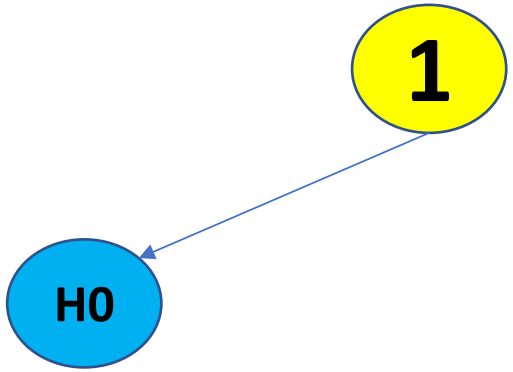


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
          for ( i = 0 ; i < procesos; i++) {
               if ( pids[i] == ret ) {
                        sprintf(buff,
                       "acaba el proceso num %d con pid %d \n" ,i,ret);
                        write(1,buff,strlen(buff));
               }
          }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

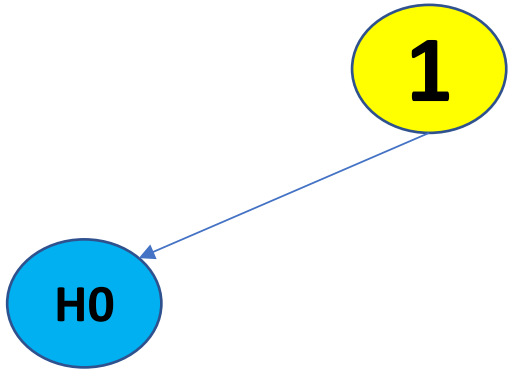


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

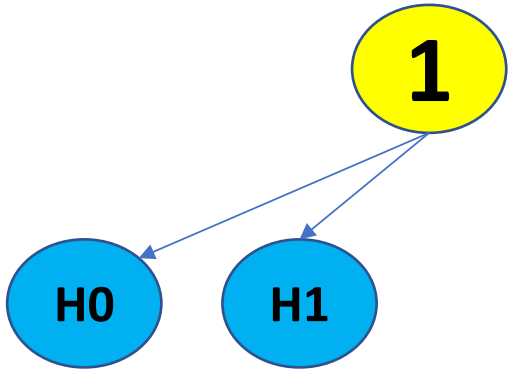


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

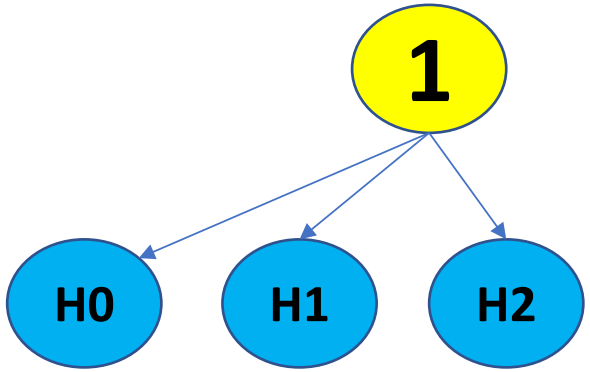


i = 1

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

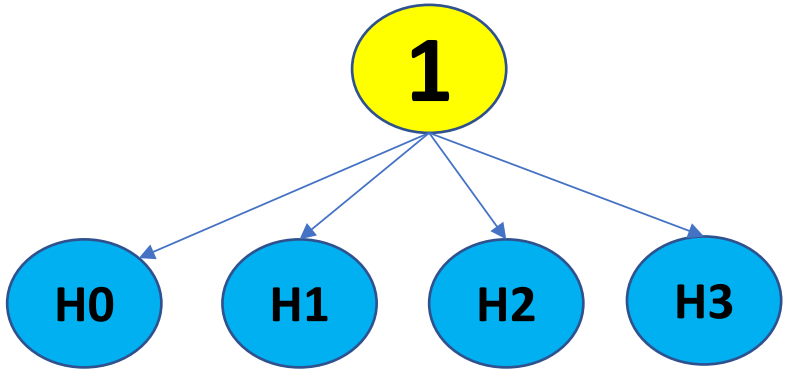


i = 2

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

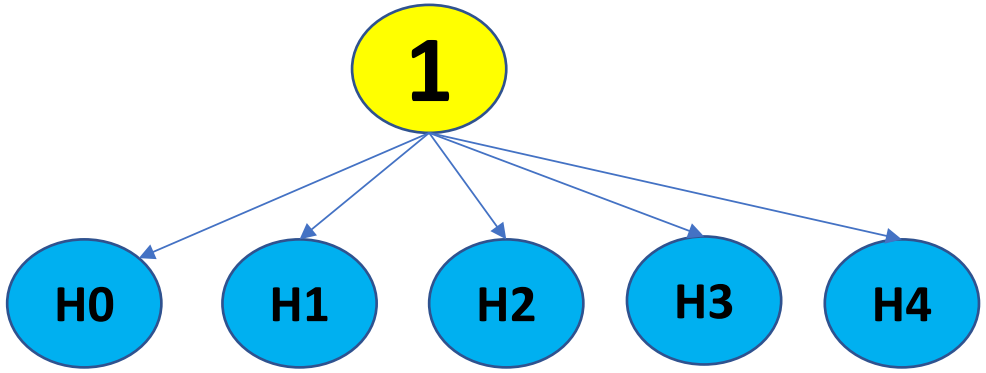



```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

i = 3

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

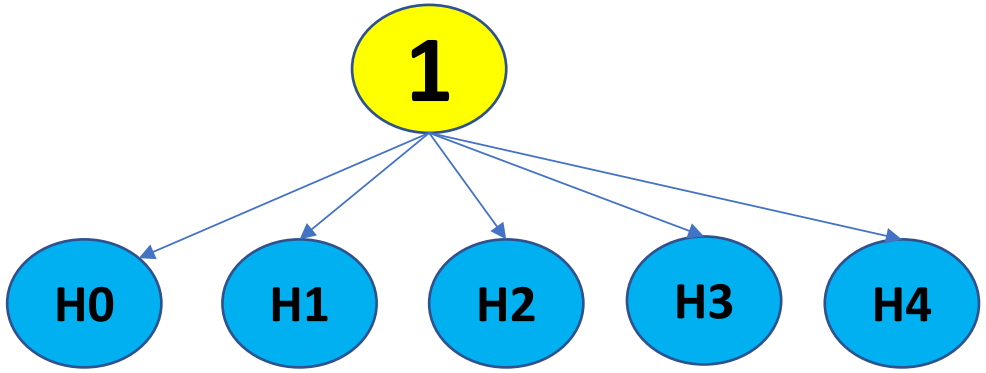


i = 4

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 0
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```



i = 5

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
         for ( i = 0 ; i < procesos; i++) {
              if ( pids[i] == ret ) {
                    sprintf(buff,
                   "acaba el proceso num %d con pid %d \n" ,i,ret);
                    write(1,buff,strlen(buff));
              }
         }
    }
}
```

1) Dibuja la jerarquía de procesos que se genera al ejecutar:
(asigna identificadores a los procesos para poder referirte después a ellos):

```
>   ./Mtarea 5 1
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
         for ( i = 0 ; i < procesos; i++) {
             if ( pids[i] == ret ) {
                 sprintf(buff,
                 "acaba el proceso num %d con pid %d \n" ,i,ret);
                 write(1,buff,strlen(buff));
             }
         }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```


```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

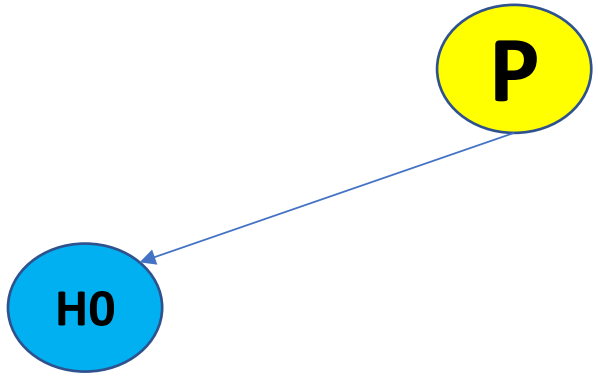
```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```


```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

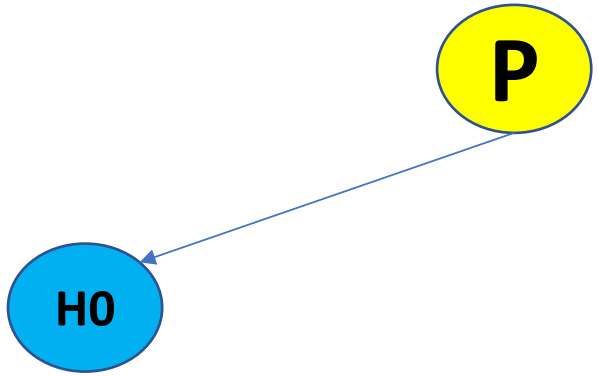


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

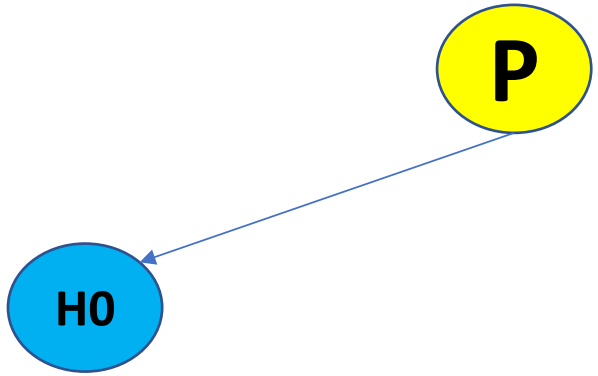


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
          for ( i = 0 ; i < procesos; i++) {
               if ( pids[i] == ret ) {
                        sprintf(buff,
                       "acaba el proceso num %d con pid %d \n" ,i,ret);
                        write(1,buff,strlen(buff));
               }
          }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

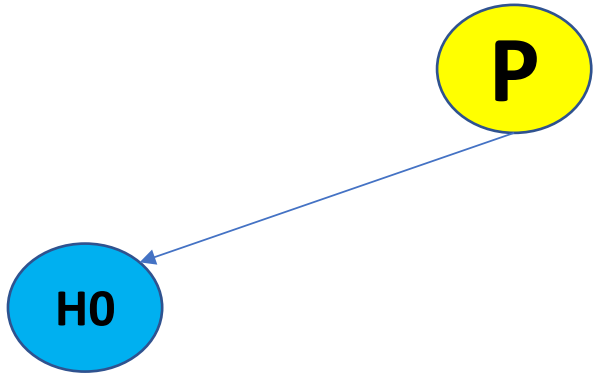


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
        it = 0;
        while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

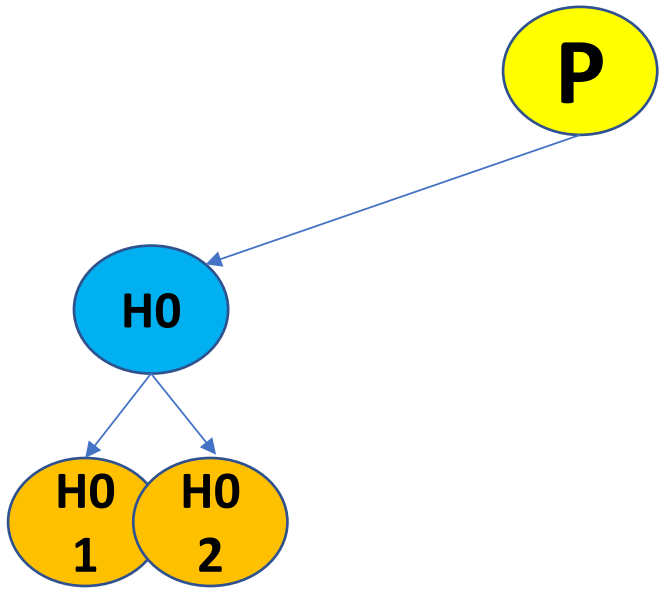


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
        while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

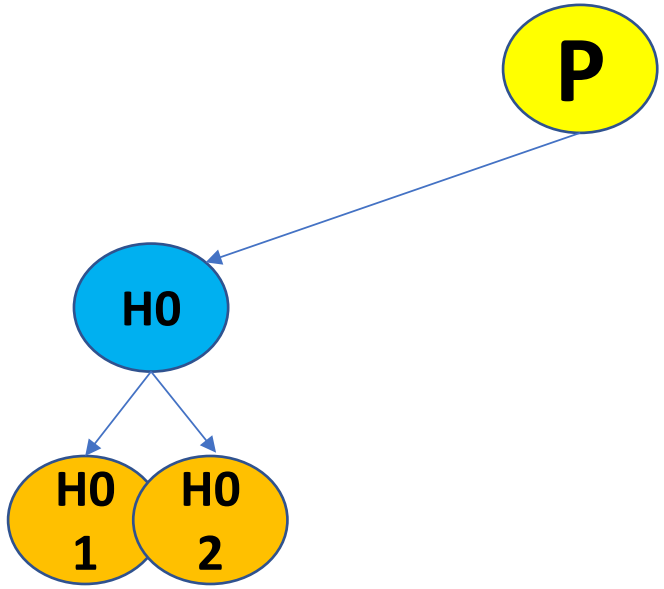


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

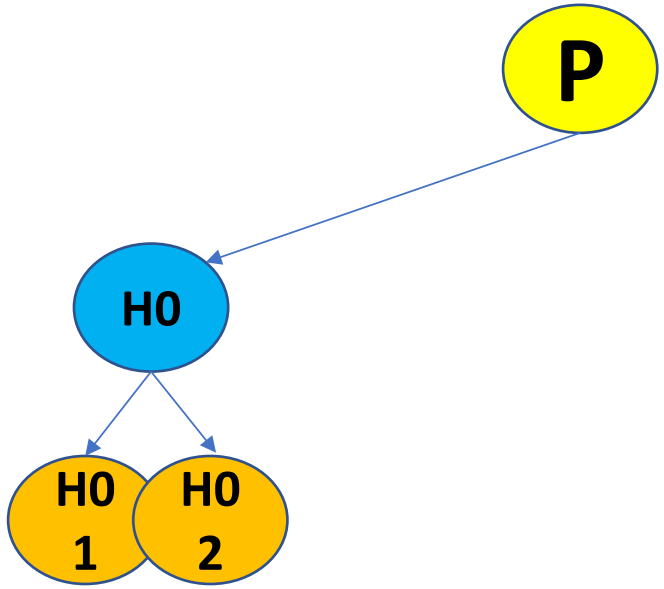


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
    }
    realizatarea (i);
    exit (1);
}
```

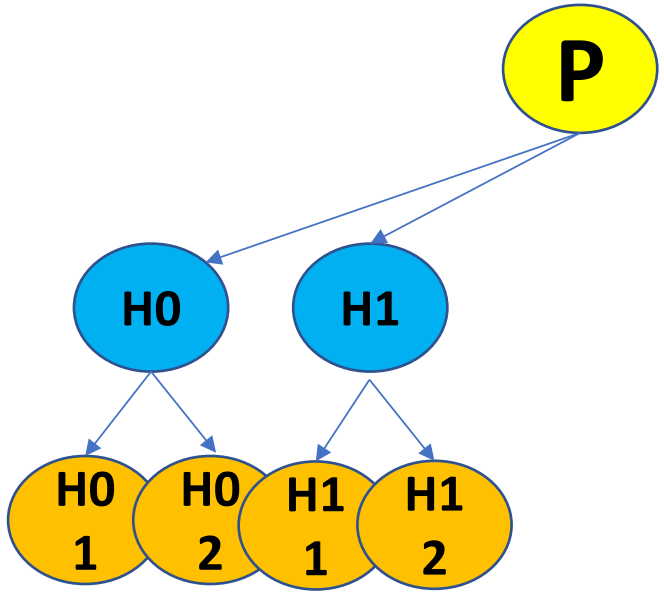


i = 0

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

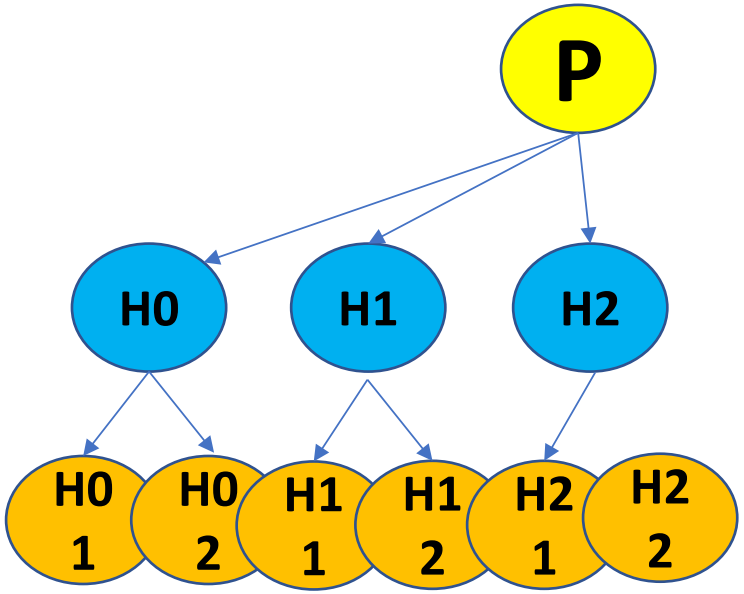


i = 1

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

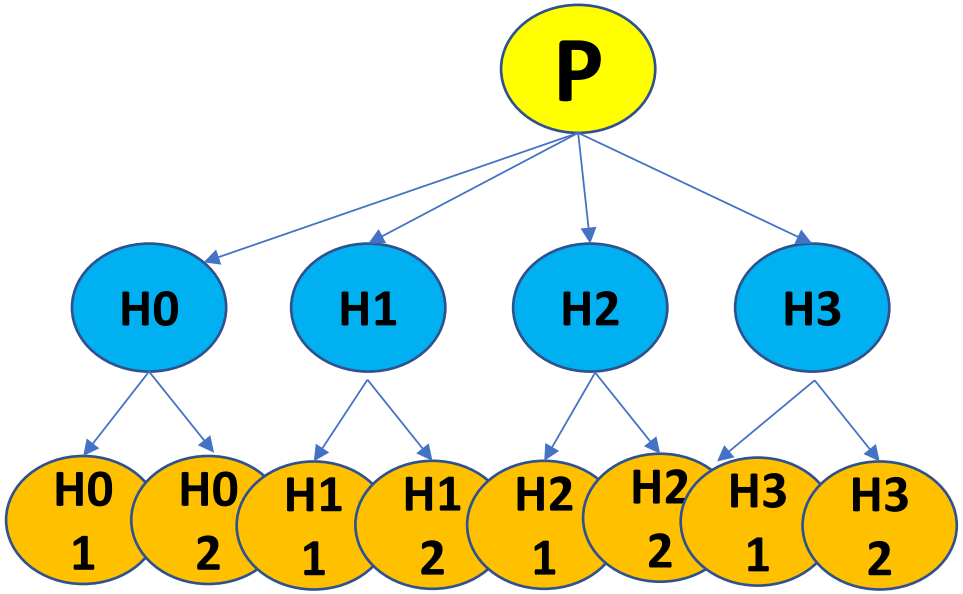


i = 2

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

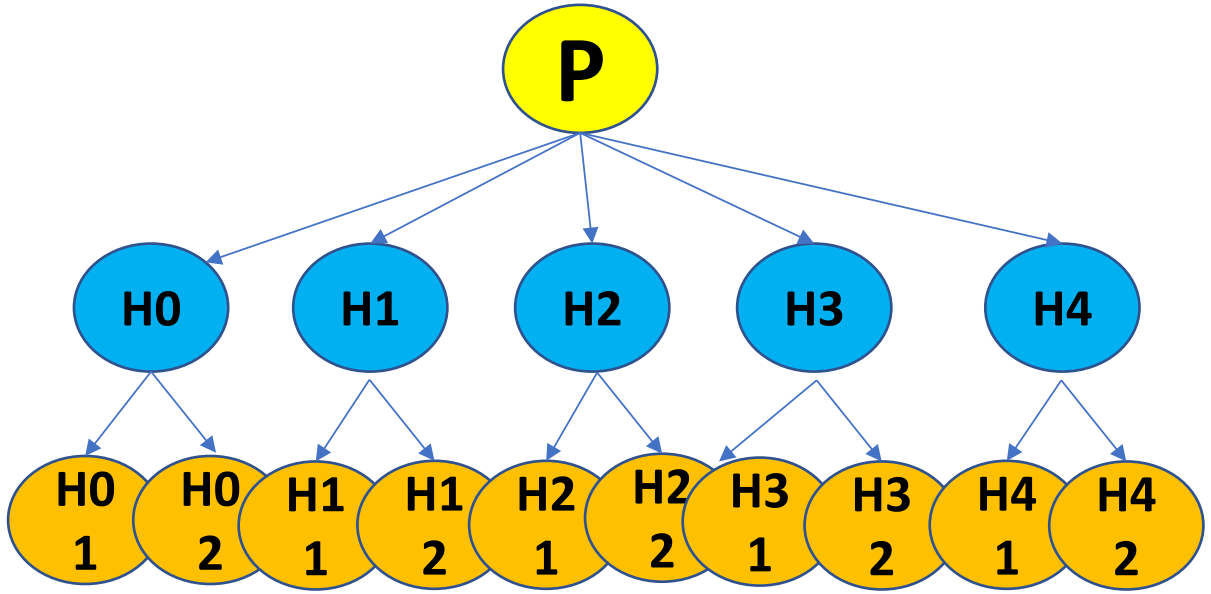


i = 3

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
    }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
        for ( i = 0 ; i < procesos; i++) {
            if ( pids[i] == ret ) {
                sprintf(buff,
                "acaba el proceso num %d con pid %d \n" ,i,ret);
                write(1,buff,strlen(buff));
            }
        }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```

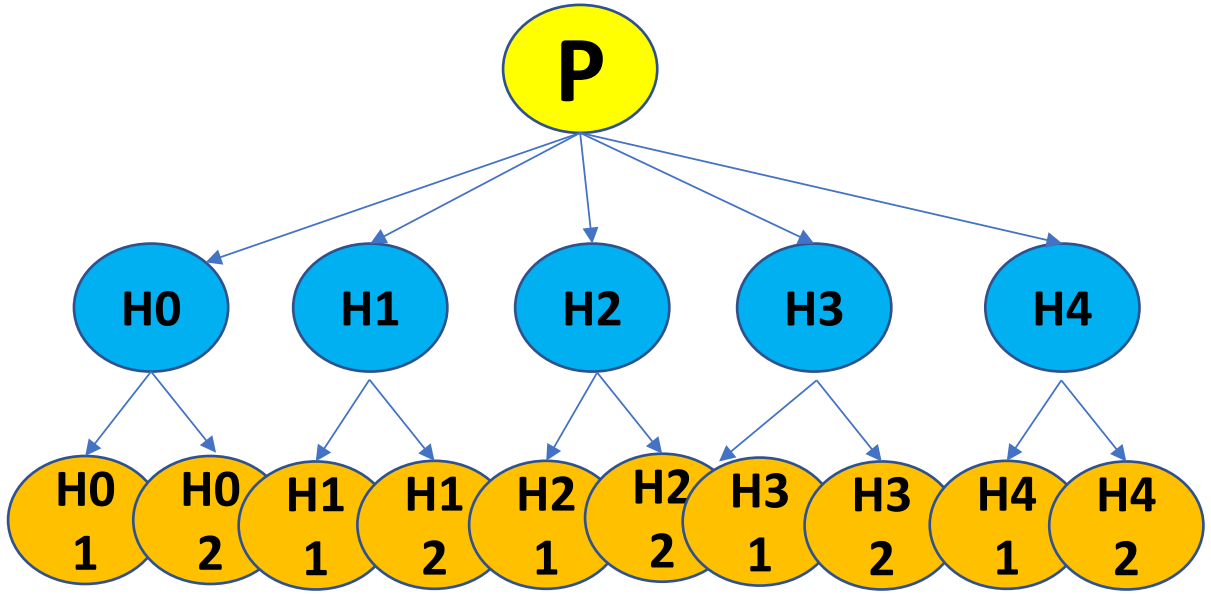


i = 4

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
     while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
          for ( i = 0 ; i < procesos; i++) {
               if ( pids[i] == ret ) {
                    sprintf(buff,
                   "acaba el proceso num %d con pid %d \n" ,i,ret);
                    write(1,buff,strlen(buff));
               }
          }
    }
}
```

```
./Mtarea 5 1
```

```
int *pids;
void usage () { … }
void realizatarea (int i) {…}
void procesardatos (int i, int multiproceso) {
    int it;
    if (multiproceso > 0) {
            it = 0;
            while ( it < 2 && fork() > 0 ) it++;
   }
   realizatarea (i);
   exit (1);
}
```



i = 5

```
void main (int argc,char *argv[]) {
    int i,ret,procesos;
    char buff[128];
    if (argc!=3) usage ();
    procesos = atoi (argv[1]);
    pids = sbrk (procesos * sizeof (int));
    for ( i = 0; i < procesos ; i++) {
        ret=fork();
        if ( ret == 0 ) procesardatos (i, atoi (argv[2]));
        pids[i] = ret;
     }
    while ( (ret = waitpid (-1,NULL,0)) > 0 ) {
         for ( i = 0 ; i < procesos; i++) {
              if ( pids[i] == ret ) {
                    sprintf(buff,
                   "acaba el proceso num %d con pid %d \n" ,i,ret);
                    write(1,buff,strlen(buff));
              }
         }
    }
}
```

Ejercicio 2 :

Será suficiente el tamaño del vector de pids que hemos reservado para gestionar los procesos hijos del proceso inicial?

```
int *pids;
void usage()
{
  char  b[128];
  sprintf(b,"./Mtarea procesosnivel1(cuantos)
procesosnivel2(0=no/1=si)\n");
  write(1,b,strlen(b));
  exit(0);
}
void realizatarea(int i){
  // Omitimos su código por simplicidad pero no hay ninguna
  // llamada a sistema relevante
  // para el ejercicio
}

void procesardatos(int i, int multiproceso)
{
   int it;
   if (multiproceso>0){ it=0;  while((fork()>0) && (it<2)) it++;}
   realizatarea(i);
   exit(1);
}
```

```
void main(int argc,char *argv[])
{
  int i,ret,procesos;
  char buff[128];
  if (argc!=3) usage();
  procesos=atoi(argv[1]);
  pids=sbrk(procesos*sizeof(int));
  for(i=0;i<procesos;i++){
   ret=fork();
    if (ret==0) procesardatos(i,atoi(argv[2]));
    pids[i]=ret;
  }
  while((ret=waitpid(-1,NULL,0))>0){
    for(i=0;i<procesos;i++){
      if (pids[i]==ret){
        sprintf(buff,
                    "acaba el proceso num %d con pid %d \n" ,i,ret);
        write(1,buff,strlen(buff));
      }
    }
  }
}
```

Ejercicio 2 : Qué procesos ejecutarán las líneas 36+37 ?

```
int *pids;
void usage()
{
  char  b[128];
  sprintf(b,"./Mtarea procesosnivel1(cuantos)
procesosnivel2(0=no/1=si)\n");
  write(1,b,strlen(b));
  exit(0);
}
void realizatarea(int i){
  // Omitimos su código por simplicidad pero no hay ninguna
  // llamada a sistema relevante
  // para el ejercicio
}

void procesardatos(int i, int multiproceso)
{
   int it;
   if (multiproceso>0){ it=0;  while((fork()>0) && (it<2)) it++;}
   realizatarea(i);
   exit(1);
}
```

```
void main(int argc,char *argv[])
{
  int i,ret,procesos;
  char buff[128];
  if (argc!=3) usage();
  procesos=atoi(argv[1]);
  pids=sbrk(procesos*sizeof(int));
  for(i=0;i<procesos;i++){
   ret=fork();
    if (ret==0) procesardatos(i,atoi(argv[2]));
    pids[i]=ret;
  }
  while((ret=waitpid(-1,NULL,0))>0){
    for(i=0;i<procesos;i++){
      if (pids[i]==ret){
        sprintf(buff,
                    "acaba el proceso num %d con pid %d \n" ,i,ret);
        write(1,buff,strlen(buff));
      }
    }
  }
}
```

Ejercicio 2 : Qué procesos ejecutarán las líneas 36+37 ? Sólo el proceso inicial, el resto acaban en el exit(0)

```
int *pids;
void usage()
{
  char  b[128];
  sprintf(b,"./Mtarea procesosnivel1(cuantos)
procesosnivel2(0=no/1=si)\n");
  write(1,b,strlen(b));
  exit(0);
}
void realizatarea(int i){
  // Omitimos su código por simplicidad pero no hay ninguna
  // llamada a sistema relevante
  // para el ejercicio
}

void procesardatos(int i, int multiproceso)
{
   int it;
   if (multiproceso>0){ it=0;  while((fork()>0) && (it<2)) it++;}
   realizatarea(i);
   exit(1);
}
```

```
void main(int argc,char *argv[])
{
  int i,ret,procesos;
  char buff[128];
  if (argc!=3) usage();
  procesos=atoi(argv[1]);
  pids=sbrk(procesos*sizeof(int));
  for(i=0;i<procesos;i++){
   ret=fork();
    if (ret==0) procesardatos(i,atoi(argv[2]));
    pids[i]=ret;
  }
  while((ret=waitpid(-1,NULL,0))>0){
    for(i=0;i<procesos;i++){
      if (pids[i]==ret){
        sprintf(buff,
                    "acaba el proceso num %d con pid %d \n" ,i,ret);
        write(1,buff,strlen(buff));
      }
    }
  }
}
```