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):

```
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));
```

```
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);
}
```

P

```
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));
```

```
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);
}
```

P

```
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));
```

```
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);
```

```
P
```

```
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++) {
i = 0 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));
```

```
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);
```

```
H0
```

```
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++)
i = 0
          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));
```

H0

```
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
          ret=fork();
```

```
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++)
      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));
```

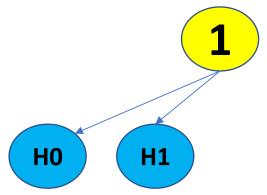
```
./Mtarea 5 0
```

H0

```
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);
```

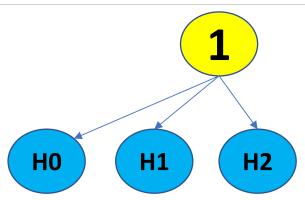
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for (i = 0; i < procesos; i++)
i = 0
          ret=fork();
           if (ret == 0) procesardatos (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));
```

```
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);
}
```



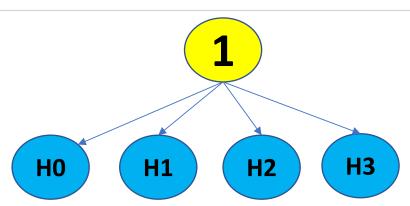
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for (i = 0; i < procesos; i++)
i = 1 ret=fork();
           if (ret == 0) procesardatos (i, atoi (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));
```

```
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);
}
```



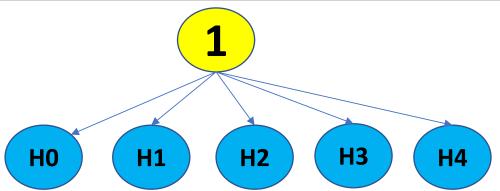
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for ( i = 0; i < procesos ; i++) {
i = 2 ret=fork();
           if (ret == 0) procesardatos (i, 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));
```

```
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++)
i = 3 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));
```

```
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++)
i = 4 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));
```

```
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);
```

```
1
H0 H1 H2 H3 H4
```

```
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++)
i = 5
           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):

```
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));
```

```
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);
}
```

P

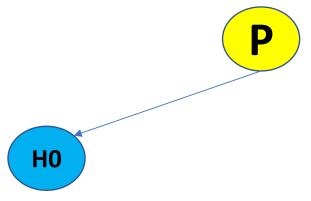
```
void main (int argc,char *argv[]) {
  int i,ret,procesos;
  char buff[128];
  if (argc!=3) usage ();
  procesos = atoi (argv[1]);
  pids = sbrk (procesos * sizeof (int));
  for (i = 0; i < procesos; i++)
      ret=fork();
      if (ret == 0) procesardatos (i, atoi (argv[2]));
      pids[i] = ret;
   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));
```

```
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);
}
```

P

```
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));
```

```
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++) {
i = 0 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));
```

```
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);
```

```
P
```

```
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++)
i = 0
          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));
```

```
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);
```

```
P
```

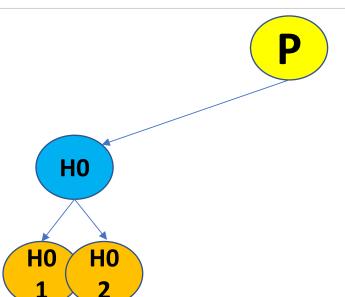
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for (i = 0; i < procesos; i++)
i = 0
          ret=fork();
           if (ret == 0) procesardatos (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));
```

```
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);
```

```
P
```

```
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++)
i = 0
          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));
```

```
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++)
i = 0
          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));
```

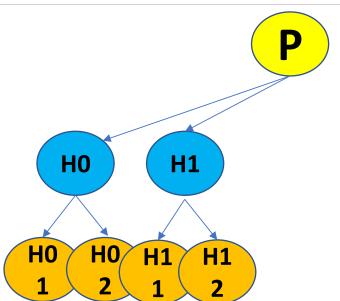
```
int *pids;
void usage () { ... }
void realizatarea (int i) {...}
void procesardatos (int i, int multiproceso) {
  int it;
  if (multiproceso > 0) {
           it = 0;
           while ( it < 2 && fork() > 0 ) it++;
                                                                 i = 0
  realizatarea (i);
  exit (1);
      H0
         H0
```

```
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));
```

```
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++;
                                                                 i = 0
  realizatarea (i);
  exit (1);
      H0
         H0
```

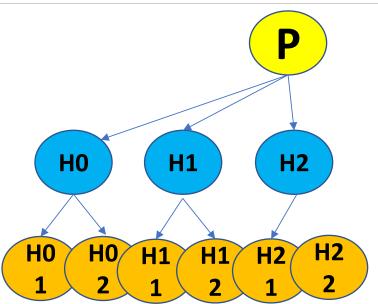
```
void main (int argc,char *argv[]) {
   int i,ret,procesos;
  char buff[128];
  if (argc!=3) usage ();
  procesos = atoi (argv[1]);
  pids = sbrk (procesos * sizeof (int));
  for (i = 0; i < procesos; i++)
      ret=fork();
      if (ret == 0) procesardatos (i, atoi (argv[2]));
      pids[i] = ret;
   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));
```

```
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);
```



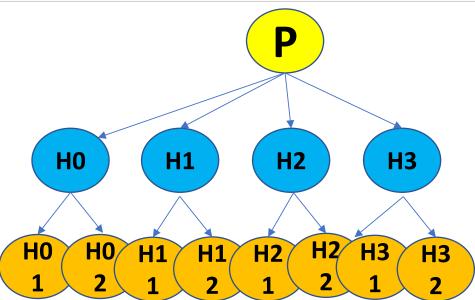
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for (i = 0; i < procesos; i++)
i = 1 ret=fork();
           if (ret == 0) procesardatos (i, atoi (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));
```

```
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);
```



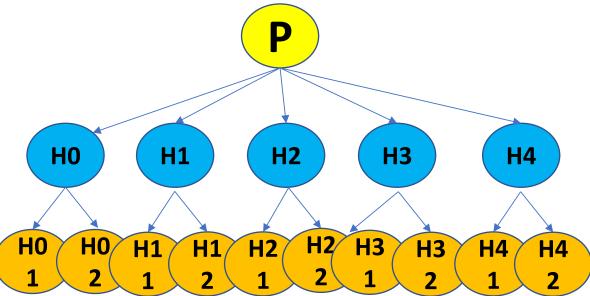
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for ( i = 0; i < procesos ; i++) {
i = 2 ret=fork();
           if ( ret == 0 ) procesardatos (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));
```

```
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);
}
```



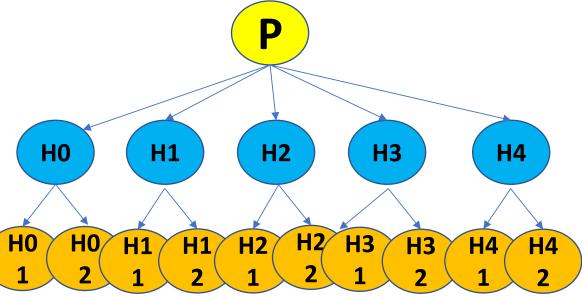
```
void main (int argc,char *argv[]) {
       int i,ret,procesos;
       char buff[128];
       if (argc!=3) usage ();
       procesos = atoi (argv[1]);
       pids = sbrk (procesos * sizeof (int));
       for (i = 0; i < procesos; i++)
i = 3 ret=fork();
          if ( ret == 0 ) procesardatos (i, atoi (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));
```

```
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++)
i = 4 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));
```

```
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++) {
i = 5
           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 main(int argc,char *argv[])
void usage()
                                                                   int i,ret,procesos;
 char b[128];
                                                                   char buff[128];
 sprintf(b,"./Mtarea procesosnivel1(cuantos)
                                                                   if (argc!=3) usage();
procesosnivel2(0=no/1=si)\n");
                                                                   procesos=atoi(argv[1]);
 write(1,b,strlen(b));
                                                                   pids=sbrk(procesos*sizeof(int));
                                                                   for(i=0;iiprocesos;i++){
 exit(0);
                                                                    ret=fork();
                                                                    if (ret==0) procesardatos(i,atoi(argv[2]));
void realizatarea(int i){
// Omitimos su código por simplicidad pero no hay ninguna
                                                                    pids[i]=ret;
// llamada a sistema relevante
                                                                   while((ret=waitpid(-1,NULL,0))>0){
// para el ejercicio
                                                                    for(i=0;iiprocesos;i++){
                                                                     if (pids[i]==ret){
void procesardatos(int i, int multiproceso)
                                                                       sprintf(buff,
                                                                                  "acaba el proceso num %d con pid %d \n",i,ret);
 int it;
                                                                       write(1,buff,strlen(buff));
 if (\text{multiproceso}>0){ it=0; while((\text{fork}()>0) && (it<2)) it++;}
 realizatarea(i);
 exit(1);
```

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

```
int *pids;
                                                                 void main(int argc,char *argv[])
void usage()
                                                                   int i,ret,procesos;
 char b[128];
                                                                   char buff[128];
 sprintf(b,"./Mtarea procesosnivel1(cuantos)
                                                                   if (argc!=3) usage();
procesosnivel2(0=no/1=si)\n");
                                                                   procesos=atoi(argv[1]);
 write(1,b,strlen(b));
                                                                   pids=sbrk(procesos*sizeof(int));
                                                              for(i=0;iiprocesos;i++){
 exit(0);
                                                              - ret=fork();
void realizatarea(int i){
                                                                    if (ret==0) procesardatos(i,atoi(argv[2]));
// Omitimos su código por simplicidad pero no hay ninguna
                                                                    pids[i]=ret;
// llamada a sistema relevante
// para el ejercicio
                                                                  while((ret=waitpid(-1,NULL,0))>0){
                                                                    for(i=0;iiprocesos;i++){
                                                                     if (pids[i]==ret){
void procesardatos(int i, int multiproceso)
                                                                      sprintf(buff,
                                                                                 "acaba el proceso num %d con pid %d \n",i,ret);
 int it;
                                                                      write(1,buff,strlen(buff));
 if (\text{multiproceso}>0){ it=0; while((\text{fork}()>0) && (it<2)) it++;}
 realizatarea(i);
 exit(1);
```

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

```
int *pids;
                                                                 void main(int argc,char *argv[])
void usage()
                                                                   int i,ret,procesos;
 char b[128];
                                                                   char buff[128];
                                                                   if (argc!=3) usage();
 sprintf(b,"./Mtarea procesosnivel1(cuantos)
procesosnivel2(0=no/1=si)\n");
                                                                   procesos=atoi(argv[1]);
 write(1,b,strlen(b));
                                                                   pids=sbrk(procesos*sizeof(int));
                                                              for(i=0;iiprocesos;i++){
 exit(0);
                                                                 ret=fork();
                                                                    if (ret==0) procesardatos(i,atoi(argv[2]));
void realizatarea(int i){
// Omitimos su código por simplicidad pero no hay ninguna
                                                                    pids[i]=ret;
// llamada a sistema relevante
                                                                   while((ret=waitpid(-1,NULL,0))>0){
// para el ejercicio
                                                                    for(i=0;iiprocesos;i++){
                                                                     if (pids[i]==ret){
void procesardatos(int i, int multiproceso)
                                                                      sprintf(buff,
                                                                                  "acaba el proceso num %d con pid %d \n",i,ret);
 int it;
                                                                      write(1,buff,strlen(buff));
 if (\text{multiproceso}>0){ it=0; while((\text{fork}()>0) && (it<2)) it++;}
 realizatarea(i);
 exit(1);
```