

LICENCIATURA EM ENGENHARIA DE SISTEMAS E INFORMÁTICA
LICENCIATURA EM MATEMÁTICA E CIÊNCIAS DA COMPUTAÇÃO

Sistemas Operativos I

Mini-teste de avaliação¹

21 de Novembro de 2005

Duração: 1h00m

Protótipos de algumas funções e chamadas ao sistema relevantes

Processos

- `pid_t fork(void);`
- `void exit(int status);`
- `pid_t wait(int *status);`
- `pid_t waitpid(pid_t pid, int *status, int options);`
- `WIFEXITED(status);`
- `WEXITSTATUS(status);`
- `int execlp(const char *file, const char *arg, ...);`

- `int execvp(const char *file, char *const argv[]);`
- `int execve(const char *file, char *const argv[], char *const envp[]);`

Sinais

- `void (*signal(int signum, void (*handler)(int)))(int);`
 - `int kill(pid_t pid, int signum);`
 - `int alarm(int seconds);`
 - `int pause(void);`
-

I

1 Caracterize um processo *zombie* de acordo com a terminologia UNIX.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

II

Analise, corrija e complete o código da página seguinte com base no seguinte enunciado:

Implemente um programa `controlador` que permita executar simultaneamente, e com limite temporal, quatro instâncias de um programa externo. O programa principal deve terminar todos os processos filhos que ainda se encontrem em execução no fim do intervalo de tempo especificado. Assuma que o programa em questão é arrancado com a seguinte linha de comando:

```
$ controlador nseg progext
```

¹Cotação — 4+16

```

1  /* includes: stdio.h, stdlib.h, unistd.h, sys/types.h, sys/wait.h, signal.h */
2
3  volatile int flag;
4  volatile pid_t pids[4];
5
6  int handler2(int signum)
7  {
8      pid_t pids[4];
9
10
11
12
13
14
15
16
17  }
18
19  void handler1(int signum)
20  {
21      pid_t res;
22      int status;
23
24      while ( (res = waitpid(-1, &status, WNOHANG)) > 0) {
25
26
27
28
29
30
31
32
33
34
35
36  }
37
38  int main(int argc, char **argv)
39  {
40      int i;
41
42      signal(handler1, SIGCHLD);
43      signal(SIGALRM, handler2);
44
45      for ( i=0; i<4; i++ ) {
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64      }
65
66
67
68
69      while ( ) {
70          pause();
71
72      }
73
74      printf("Fim: todos os processos filho terminaram\n");
75      return 0;
76  }

```