

jobcontrol.c.pdf



macodecena



Sistemas Operativos



2º Grado en Ingeniería del Software



Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

coches.net

coches.net



♥
No pierdas
ni un minuto
✓

**En este momento hay alguien
teniendo relaciones sexuales
dentro de un coche.**

Y no eres tú. Pero podrías serlo.





27/6/23, 16:51

job_control.c

~\Desktop\ING DEL SOFTWARE\SOFTWARE 2_2\SISTEMAS
OPERATIVOS\Prácticas\Práctica 4\prShellBasico\job_control.c

```

1  /*-----
2  UNIX Shell Project
3  job_control module
4
5  Sistemas Operativos
6  Grados I. Informatica, Computadores & Software
7  Dept. Arquitectura de Computadores - UMA
8
9  Some code adapted from "Fundamentos de Sistemas Operativos", Silberschatz et al.
10 -----*/
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <string.h>
15 #include <malloc.h>
16 #include "job_control.h"
17
18 // -----
19 // get_command() reads in the next command line, separating it into distinct tokens
20 // using whitespace as delimiters. setup() sets the args parameter as a
21 // null-terminated string.
22 // -----
23
24 void get_command(char inputBuffer[], int size, char *args[], int *background)
25 {
26     int length, /* # of characters in the command line */
27         i,      /* loop index for accessing inputBuffer array */
28         start,  /* index where beginning of next command parameter is */
29         ct;     /* index of where to place the next parameter into args[] */
30
31     ct = 0;
32     *background=0;
33
34     /* read what the user enters on the command line */
35     length = read(STDIN_FILENO, inputBuffer, size);
36
37     start = -1;
38     if (length == 0)
39     {
40         printf("\nBye\n");
41         exit(0); /* ^d was entered, end of user command stream */
42     }
43     if (length < 0){
44         perror("error reading the command");
45         exit(-1); /* terminate with error code of -1 */
46     }
47
48     /* examine every character in the inputBuffer */
49     for (i=0; i<length; i++)
50     {
51         switch (inputBuffer[i])
52         {
53             case ' ':
54             case '\t': /* argument separators */
55                 if (start != -1)

```



coches.net



No pierdas
ni un minuto

```

56     {
57         args[ct] = &inputBuffer[start];    /* set up pointer */
58         ct++;
59     }
60     inputBuffer[i] = '\0'; /* add a null char; make a C string */
61     start = -1;
62     break;
63
64     case '\n':                /* should be the final char examined */
65         if (start != -1)
66         {
67             args[ct] = &inputBuffer[start];
68             ct++;
69         }
70         inputBuffer[i] = '\0';
71         args[ct] = NULL; /* no more arguments to this command */
72         break;
73
74     default :                /* some other character */
75
76         if (inputBuffer[i] == '&') // background indicator
77         {
78             *background = 1;
79             if (start != -1)
80             {
81                 args[ct] = &inputBuffer[start];
82                 ct++;
83             }
84             inputBuffer[i] = '\0';
85             args[ct] = NULL; /* no more arguments to this command */
86             i=length; // make sure the for loop ends now
87
88         }
89         else if (start == -1) start = i; // start of new argument
90     } // end switch
91 } // end for
92 args[ct] = NULL; /* just in case the input line was > MAXLINE */
93 }
94
95
96 // -----
97 /* devuelve puntero a un nodo con sus valores inicializados,
98 devuelve NULL si no pudo realizarse la reserva de memoria*/
99 job * new_job(pid_t pid, const char * command, enum job_state state)
100 {
101     job * aux;
102     aux=(job *) malloc(sizeof(job));
103     aux->pgid=pid;
104     aux->state=state;
105     aux->command=strdup(command);
106     aux->next=NULL;
107     return aux;
108 }
109
110 // -----
111 /* inserta elemento en la cabeza de la lista */
112 void add_job (job * list, job * item)
113 {
114     job * aux=list->next;
115     list->next=item;

```

coches.net

**En este momento
hay alguien
teniendo relaciones
sexuales dentro de
un coche.**

**Y no eres tú.
Pero podrías serlo.**



coches.net



♡
No pierdas
ni un minuto
↗

Sistemas Operativos



Comparte estos flyers en tu clase y consigue más dinero y recompensas



Banco de apuntes de la

WUOLAH

1

Imprime esta hoja

2

Recorta por la mitad

3

Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes

4

Llévate dinero por cada descarga de los documentos descargados a través de tu QR




```

116     item->next=aux;
117     list->pgid++;
118
119 }
120
121 // -----
122 /* elimina el elemento indicado de la lista
123 devuelve 0 si no pudo realizarse con exito */
124 int delete_job(job * list, job * item)
125 {
126     job * aux=list;
127     while(aux->next!= NULL && aux->next!= item) aux=aux->next;
128     if(aux->next)
129     {
130         aux->next=item->next;
131         free(item->command);
132         free(item);
133         list->pgid--;
134         return 1;
135     }
136     else
137         return 0;
138
139 }
140 // -----
141 /* busca y devuelve un elemento de la lista cuyo pid coincida con el indicado,
142 devuelve NULL si no lo encuentra */
143 job * get_item_bypid (job * list, pid_t pid)
144 {
145     job * aux=list;
146     while(aux->next!= NULL && aux->next->pgid != pid) aux=aux->next;
147     return aux->next;
148 }
149 // -----
150 job * get_item_bypos( job * list, int n)
151 {
152     job * aux=list;
153     if(n<1 || n>list->pgid) return NULL;
154     n--;
155     while(aux->next!= NULL && n) { aux=aux->next; n--;}
156     return aux->next;
157 }
158
159 // -----
160 /*imprime una linea en el terminal con los datos del elemento: pid, nombre ... */
161 void print_item(job * item)
162 {
163
164     printf("pid: %d, command: %s, state: %s\n", item->pgid, item->command,
165 state_strings[item->state]);
166 }
167 // -----
168 /*recorre la lista y le aplica la funcion pintar a cada elemento */
169 void print_list(job * list, void (*print)(job *))
170 {
171     int n=1;
172     job * aux=list;
173     printf("Contents of %s:\n",list->command);
174     while(aux->next!= NULL)

```



27/6/23, 16:51

job_control.c

```

175     {
176         printf(" [%d] ",n);
177         print(aux->next);
178         n++;
179         aux=aux->next;
180     }
181 }
182
183 // -----
184 /* interpretar valor status que devuelve wait */
185 enum status analyze_status(int status, int *info)
186 {
187     // el proceso se ha suspendido
188     if (WIFSTOPPED (status))
189     {
190         *info=WSTOPSIG(status);
191         return(SUSPENDED);
192     }
193     // el proceso se ha reanudado
194     else if (WIFCONTINUED(status))
195     {
196         *info=0;
197         return(CONTINUED);
198     }
199     else
200     {
201         // el proceso ha terminado
202         if (WIFSIGNALED (status))
203         {
204             *info=WTERMSIG (status);
205             return(SIGNALED);
206         }
207         else
208         {
209             *info=WEXITSTATUS(status);
210             return(EXITED);
211         }
212     }
213     return -1;
214 }
215
216 // -----
217 // cambia la accion de las seÑales relacionadas con el terminal
218 void terminal_signals(void (*func) (int))
219 {
220     signal (SIGINT, func); // ctrl+c interrupt tecleado en el terminal
221     signal (SIGQUIT, func); // ctrl+\ quit tecleado en el terminal
222     signal (SIGTSTP, func); // ctrl+z Stop tecleado en el terminal
223     signal (SIGTTIN, func); // proceso en segundo plano quiere leer del terminal
224     signal (SIGTTOU, func); // proceso en segundo plano quiere escribir en el terminal
225 }
226
227 // -----
228 void block_signal(int signal, int block)
229 {
230     /* declara e inicializa máscara */
231     sigset_t block_sigchld;
232     sigemptyset(&block_sigchld );
233     sigaddset(&block_sigchld,signal);
234     if(block)

```

localhost:60205/8ae07a40-c990-4424-b3cb-6f6cb5d2c7db/

4/5



coches.net



No pierdas
ni un minuto

```
235     {
236         /* bloquea señal */
237         sigprocmask(SIG_BLOCK, &block_sigchld, NULL);
238     }
239     else
240     {
241         /* desbloquea señal */
242         sigprocmask(SIG_UNBLOCK, &block_sigchld, NULL);
243     }
244 }
245
246
247
```