/\*

\* parse.c : use whitespace to tokenise a line

\* Initialise a vector big enough

\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include "shell.h"

/\* Parse a commandline string into an argv array. \*/

char \*\* parse(char \*line) {

static char delim[] = " \t\n"; // SPACE or TAB or NL

int count = 0;

char \* token;

char \*\*newArgv;

//Nothing entered.

if (line == NULL) {

return NULL;

}

/\* Init strtok with commandline, then get first token.

\* Return NULL if no tokens in line.

\*

\* Fill in code.

\*/

token = strtok(line,delim);

if (token == NULL) {

return NULL;

}

/\* Create array with room for first token.

\*

\* Fill in code.

\*/

newArgv = (char\*\*)malloc(1\* sizeof(char\*));

newArgv[0] = (char\*)malloc(strlen(token)\*sizeof(char));

strcpy(newArgv[0],token);

/\* While there are more tokens...

\*

\* - Get next token.

\* - Resize array.

\* - Give token its own memory, then install it.

\*

\* Fill in code.

\*/

while (token != NULL){

token = strtok (NULL, delim);

if(token == NULL) break;

newArgv = (char\*\*) realloc(newArgv,((count+2)\*sizeof(char\*)));

newArgv[count+1] = (char\*)malloc(strlen(token)\*sizeof(char));

strcpy(newArgv[count+1],token);

count++;

}

/\* Null terminate the array and return it.

\*

\* Fill in code.

\*/

newArgv = (char\*\*) realloc(newArgv,((count+2)\*sizeof(char\*))); // 應該+1就可以了

newArgv[count+1] = NULL;

int y=0;

while( newArgv[y]!=NULL ){

if(newArgv[y]==NULL) break;

printf("[%d] : %s \n",y,newArgv[y]);

y++;

}

return newArgv;

}

/\*

\* Free memory associated with argv array passed in.

\* Argv array is assumed created with parse() above.

\*/

void free\_argv(char \*\*oldArgv) {

/\* Free each string hanging off the array.

\* Free the oldArgv array itself.

\*

\*Fill in code.

\*/

int i=0;

while(oldArgv[i] !=NULL )

{

if(oldArgv[i] == NULL)break;

free(oldArgv[i]);

i++;

}

free(oldArgv);

}