Sistemas Operativos

Mendez-Simó

Lab Shell

Entrega Challenges
(solo multiple pipes)

11/05

Hernán Tain

parsing.c

```
struct cmd* aux_func(char* buf, char* right) {

      struct cmd * pipeexecs[20];
      char* pipeargs[20];
      char* a_right;
      int i = 0, j, idx;
      pipeargs[i] = buf;

      while((idx = block_contains(right, '|' )) > 0){
            i++;
            a_right = split_line(right, '|' );
            pipeargs[i] = right;
            right = a_right;
      }

      i++;
      pipeargs[i] = right;

      for( j = 0; j <= i; j++ ) {
            pipeexecs[j] = parse_cmd(pipeargs[j]);
      }
      pipeexecs[j++] = NULL;

      return multipipe_cmd_create( pipeexecs, j-1 );
}

// parses the command line
// looking for the pipe character '|'
struct cmd* parse_line(char* buf) {

      struct cmd *r, *l;
      char* right = split_line(buf, '|');
      int idx;

      if((idx = block_contains(right, '|')) > 0)
            return aux_func(buf,right);

      l = parse_cmd(buf);
      r = parse_cmd(right);

      return pipe_cmd_create(l, r);
}
```

exec.c

```
int multipipe_func (int in, int out, struct command *cmd) {

      struct execcmd* e;
      pid_t pid;

      if ((pid = fork ()) == 0){

      if (in != 0) {
            dup2(in, 0);
            close(in);
            }

            if (out != 1) {
            dup2(out, 1);
            close(out);
            }

            e = (struct execcmd*) cmd;

            return execvp(e->argv[0], e->argv);
    }

      return pid;
}
```

exec.c

Dentro del switch del case

```
            case MULTIPIPE: {

                  struct multipipecmd* mp = (struct multipipecmd*) cmd;

                  int i;
                  pid_t pid;
                  int in = 0, fd[2];

                  for(i = 0; i < mp->argc - 1; i++){
                        pipe(fd);
                        multipipe_func( in,fd[WRITE], mp->commands[i] );
                        close(fd[1]);
                        in = fd[0];
                  }

                  if(in != 0)
                        dup2(in,0);

                  c = (struct execmd*) mp->commands[i];
                  execvp(c->argv[0],c->argv);
            }
      }
}
```

createcmd.c

```
struct cmd* multipipe_cmd_create(struct cmd** array, int num_args) {

      struct multipipecmd* mp;

      mp = (struct multipipecmd*)calloc( sizeof(*mp), sizeof(*mp) );

      mp->type = MULTIPIPE;

      int i;
      for(i = 0; i <= num_args; i++){
            mp->commands[i] = array[i];
      }

      mp->commands[i++] = NULL;
      mp->argc = num_args;

      return (struct cmd*) mp;
}
```

types.h

```
struct multipipecmd {
      int type;
      pid_t pid;
      struct cmd* commands[MAXARGS];
      int argc;
};
```