# COMP 7500/7506 Lecture 19: Project 3-7 Command-Line Parser

/\* The run command - submit a job. \*/

int cmd\_run(int nargs, char \*\*args) {

if (nargs != 4) {

printf("Usage: run <job> <time> <priority>\n");

return EINVAL;

}

/\* Use execv to run the submitted job in AUbatch \*/

printf("use execv to run the job in AUbatch.\n");

return 0; /\* if succeed \*/

}

/\* Display menu information \*/

void showmenu(const char \*name, const char \*x[]) {

int ct, half, i;

printf("\n");

printf("%s\n", name);

for (i=ct=0; x[i]; i++) {

ct++;

}

half = (ct+1)/2;

for (i=0; i<half; i++) {

printf(" %-36s", x[i]);

if (i+half < ct) {

printf("%s", x[i+half]);

}

printf("\n");

}

printf("\n");

}

static const char \*helpmenu[] = {

"[run] <job> <time> <priority> ",

"[quit] Exit AUbatch ",

"[help] Print help menu ",

NULL /\* Please add more menu options below \*/

};

int cmd\_helpmenu(int n, char \*\*a) {

(void)n;

(void)a;

showmenu("AUbatch help menu", helpmenu);

return 0;

}

/\* Command table. \*/

static struct {

const char \*name;

int (\*func)(int nargs, char \*\*args);

} cmdtable[] = {

/\* commands: single command must end with \n \*/

{ "?\n", cmd\_helpmenu },

{ "h\n", cmd\_helpmenu },

{ "help\n", cmd\_helpmenu },

{ "r", cmd\_run },

{ "run", cmd\_run },

{ "q\n", cmd\_quit },

{ "quit\n", cmd\_quit },

{NULL, NULL} /\* Please add more operations below. \*/

};

/\* Process a single command. \*/

int cmd\_dispatch(char \*cmd) {

time\_t beforesecs, aftersecs, secs;

u\_int32\_t beforensecs, afternsecs, nsecs;

char \*args[MAXMENUARGS];

int nargs=0;

char \*word;

char \*context;

int i, result;

for (word = strtok\_r(cmd, " ", &context);

word != NULL;

word = strtok\_r(NULL, " ", &context)) {

if (nargs >= MAXMENUARGS) {

printf("Command line has too many words\n");

return E2BIG;

}

args[nargs++] = word;

}

if (nargs==0) {

return 0;

}

for (i=0; cmdtable[i].name; i++) {

if (\*cmdtable[i].name && !strcmp(args[0], cmdtable[i].name)) {

assert(cmdtable[i].func!=NULL);

result = cmdtable[i].func(nargs, args);

return result;

}

}

printf("%s: Command not found\n", args[0]);

return EINVAL;

}

int main() {

char \*buffer;

size\_t bufsize = 64;

buffer = (char\*) malloc(bufsize \* sizeof(char));

if (buffer == NULL) {

perror("Unable to malloc buffer");

exit(1);

}

while (1) {

printf("> [? for menu]: ");

getline(&buffer, &bufsize, stdin);

cmd\_dispatch(buffer);

}

return 0;

}