Task 1:

Text

Description automatically generated

Task 2:

Text

Description automatically generated

Shell1.c Listing:

/\*

\* shell1.c

\* simplest shell.

\* running in loop to read input string (command) to be processed

\* To exit, type EOF (ctlr+D) or ctlr+C

\*/

#include <sys/types.h>

#include <sys/wait.h>

#include <errno.h>

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sysexits.h>

#include <unistd.h>

**char** \*getinput(**char** \*buffer, size\_t buflen)

{

printf("$$ ");

**return** fgets(buffer, buflen, stdin);

}

**int** main(**int** argc, **char** \*\*argv)

{

**char** buf[1024];

pid\_t pid;

**int** status;

**while** (getinput(buf, **sizeof**(buf))) {

buf[strlen(buf) - 1] = '\0';

**if**(strcmp(buf, "exit") == 0) {

exit(1);

}

**if**((pid=fork()) == -1) {

fprintf(stderr, "shell: can't fork: %s\n",

strerror(errno));

**continue**;

} **else** **if** (pid == 0) {

/\* child \*/

execlp(buf, buf, (**char** \*)0);

fprintf(stderr, "shell: couldn't exec %s: %s\n", buf,

strerror(errno));

exit(EX\_DATAERR);

}

**if** ((pid=waitpid(pid, &status, 0)) < 0)

fprintf(stderr, "shell: waitpid error: %s\n",

strerror(errno));

}

exit(EX\_OK);

}

Task 3:

Text

Description automatically generated

Shell2.c Listing:

/\*

\* shell2.c

\* simplest shell.

\* running in loop to read input string (command) to be processed

\* To exit, type EOF (ctlr+D) or ctlr+C

\*/

#include <sys/types.h>

#include <sys/wait.h>

#include <errno.h>

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sysexits.h>

#include <unistd.h>

**char** \*getinput(**char** \*buffer, size\_t buflen)

{

printf("$$ ");

**return** fgets(buffer, buflen, stdin);

}

**void** sig\_int(**int** signo) {

printf("\nCaught SIGINT!\n");

}

**int** main(**int** argc, **char** \*\*argv)

{

**char** buf[1024];

pid\_t pid;

**int** status;

signal(SIGINT, sig\_int);

**while** (getinput(buf, **sizeof**(buf))) {

buf[strlen(buf) - 1] = '\0';

// handles if user input is "exit"

**if**(strcmp(buf, "exit") == 0) {

exit(1);

}

**if**((pid=fork()) == -1) {

fprintf(stderr, "shell: can't fork: %s\n",

strerror(errno));

**continue**;

} **else** **if** (pid == 0) {

/\* child \*/

execlp(buf, buf, (**char** \*)0);

fprintf(stderr, "shell: couldn't exec %s: %s\n", buf,

strerror(errno));

exit(EX\_DATAERR);

}

**if** ((pid=waitpid(pid, &status, 0)) < 0)

fprintf(stderr, "shell: waitpid error: %s\n",

strerror(errno));

}

exit(EX\_OK);

}