1. touch .hello_there
2. chmod 000 .hello_there
3. Ctrl+Z → bg(possibly fg too, needs a deeper look)
4. export ANSWER=42
5. mkfifo magic_mirror
6. this exe:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>

int main(){
        int file_desc = open("ch5.txt", O_CREAT | O_RDWR);
        dup2(file_desc, 99);
        if(fork() == 0){
                char* argv[] = {NULL,NULL,NULL};
                char* envp[] = {NULL,NULL,NULL};
                if(execve("./riddle",argv,envp) == -1){
                        perror("Could not execve");
                }
        }
        return 0;

}
```

7. this exe:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>

int main(){
        int fds1[2];
        if(pipe(fds1) < 0){
                perror("Failed to Create Pipe");
                return 1;
        }
        dup2(fds1[0],33);
        dup2(fds1[1],1);
        dup2(fds1[1],34);
        dup2(fds1[0],53);
        dup2(fds1[1],54);
        if(fork() == 0){
                char* argv[] = {NULL,NULL,NULL};
                char* envp[] = {NULL,NULL,NULL};
                if(execve("./riddle",argv,envp) == -1){
                        perror("Could not execve");
                }
        }
        return 0;
}
```

8. ln .hello_there .hey_there (make a second hard link to the same file)
9. this exe and script:

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include <string.h>
int main(int argc, char *argv[]){
        if (argc != 2){
                printf("Exactly one file should be given");
                return 1;
        }
        int bf_file = open(argv[1], O_WRONLY);
        lseek(bf_file,1073741820,SEEK_SET);
        write(bf_file,"111111111111111111111111",strlen("111111111111111111111111"));
        close(bf_file);
        return 0;
}
```

```sh
#!/bin/sh

for n in $(seq 0 9);
do
        filename=$(echo bf0$(echo $n));
        dd if=/dev/zero of=$filename bs=1 count=0 seek=1500M;
        ./ch8writer $filename;
done
```

10. ./ch9server [> testfile] & ; ./riddle

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
int main(int argc, char *argv[]){
    int sockfd,newsockfd,portno,clilen;
    char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;
    int n;
    char first_num_string[5], string_to_send[5];
    int first_num, num_to_send;

    portno = 49842;
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1){
        perror("Failed to create socket");
        return 1;
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(portno);
    serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) == -1){
        perror("Failed to bind socket");
        return 1;
    }
```

```
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0){
        perror("Failed to accept connection");
        return 1;
    }
    bzero(buffer,256);
    n = read(newsockfd,buffer,255);
    if (n < 0){
        perror("Failed to read from socket");
        return 1;
    }
    memcpy(first_num_string,&buffer[12],5);
    first_num_string[4] = '\0';
    first_num = atoi(first_num_string);
    num_to_send = first_num + 1;
    sprintf(string_to_send,"%d",num_to_send);
    n = write(newsockfd,string_to_send,5);
    printf("Here's what I extracted: %s\n",first_num_string);
    return 0;
}
```

11. touch secret_number; ln secret_number spy_file; ./riddle &; cat spy_file; fg; [number read]
12. touch secret_number; exec 3<secret_number;./riddle &; read -u 3 a; echo $a; fg; [number read] or a second shell
13. ./riddle; Ctrl-Z; ./ch12memwriter [pid] [memaddr] [char]; fg

```
int main(int argc, char* argv[]){
    if ( argc != 4 ){
        printf("Three cmd args needed. PID memaddr char");
    }
    long int pid = atol(argv[1]);
    long int memaddr = strtol(argv[2],NULL,16);
    char to_write = argv[3][0];
    char file_name[64];
    sprintf(file_name, "/proc/%ld/mem", pid);

    int mem_file = open(file_name, O_RDWR);

    ptrace(PTRACE_ATTACH,pid,0,0);
    waitpid(pid,NULL,0);
    if(pwrite(mem_file,&to_write,1,memaddr) == -1){
        perror("Failed To Write");
        return 1;
    }
    ptrace(PTRACE_DETACH,pid,0,0);

    close(mem_file);
    return 0;

}
```

14. ./riddle; Ctrl-Z; ./ch13memrestore ; fg

```c
int main(){
    int file = open(".hello_there", O_RDWR);
    ftruncate(file,32768);
    return 0;
}
```

15. ./ch15fork

```c
int main(){
    for(;;){
        int pid;
        pid = fork();
        if( pid == 0 ){
            /* child process */
            int my_pid = getpid();
            if(my_pid == 32767 || my_pid == 32767){
               char* argv[] = {NULL,NULL,NULL};
               char* envp[] = {NULL,NULL,NULL};
               execve ("riddle",argv,envp);
            } else{
                exit(0);
            }
        }
        if (pid == 33000){
            return 1;
        }
    }
}
```

16.