1 Desafio

No desafio, vamos modificar o *driver* e o programa principal para capturar uma linha digitada e não apenas um caractere.

Vamos desenvolver duas novas funções para o *driver*:

- `kgetc` que pode ser chamada do programa principal para capturar uma entrada tipo `char` do teclado.
- `kgets` que usa `kgetc` e pode ser chamada do programa principal para capturar uma entrada tipo `string` do teclado.

Na função `kgetc` vale chamar atenção aos cuidados tomados para evitar condições de corrida no acesso ao *buffer*. Temos a desabilitação de interrupções comentada na atividade 1 e o uso do **mutex** `kp->data` que mantém o *driver* em *busy wait* caso necessário.

```
1  int kgetc()
2  {
3    char c;
4    KBD *kp = &kbd;
5
6    unlock();                          // unmask IRQ in case it was
       masked out
7    while(kp->data == 0);              // BUSY wait while kp->data is
       0
8
9    lock();                            // mask out IRQ
10   c = kp->buf[kp->tail++];
11   kp->tail %= 128;                 /*** Critical Region ***/
12   kp->data--; kp->room++;
13   unlock();                        // unmask IRQ
14   return c;
15 }
16
17
18 int kgets(char s[ ])
19 {
20   char c;
21   while( (c = kgetc()) != '\r'){
22     kputc(c);
23     *s = c;
24     s++;
25   }
26   *s = 0;
27 }
```

Vamos modificar também `kbd_handler` para deixar de imprimir o caractere logo após a interrupção. Passaremos a realizar essa impressão no *loop* de `kgets` como pode ser visto no código acima.

Para o programa principal basta modificar o *loop* onde aguardavamos a entrada de caracter e ativamente pedir por uma nova linha a cada iteração com a função `kgets`.

```
1   while(1){
2       color = CYAN;
3       kputs("Digite uma linha: ");
4       kputs(kgets(texto));
5       kputs("\n");
6   }
```

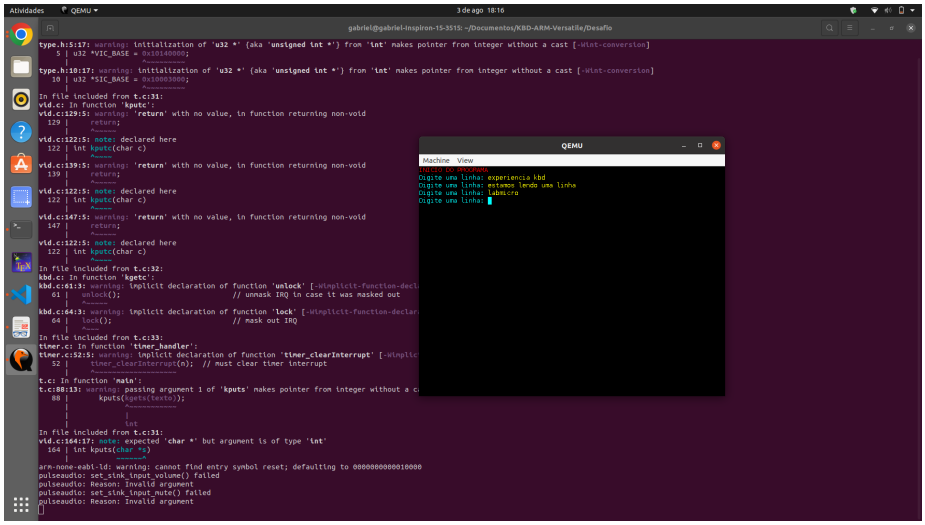

Figura 1: Execução do desafio

Para conferir a organização dos arquivos do repositório após essa etapa, consultar `https://github.com/ArkhamKnightGPC/KBD-ARM-Versatile/Desafio`.