# Sistemas Operacioanis – 2021
# Primeiro Trabalho
# Interpretador de Comandos

## 1 Descrição do Trabalho

Neste trabalho, você implementará um interpretador de comandos (*shell*) a ser executado no sistema operacional Linux. A implementação deve ser em C ou C++.

Um interpretador de comandos é um programa que recebe comandos de um usuário e os executa. Além disso, ele pode conter comandos internos para, por exemplo, definir variáveis de ambiente, mostrar o histórico de comandos, etc.

O interpretador de comandos deve conter as seguintes funcionalidades mínimas:

1. Execução de um programa com ou sem argumentos tanto em *foreground* quanto em *background* quando terminado pelo caractere &.
2. O mesmo que 1, mas com redirecionamento das saídas padrão (>) ou de erro (2>) para arquivos.
3. O mesmo que 1, mas com redirecionamento da entrada padrão (<) para um arquivo de entrada.
4. O mesmo que 1, mas com redirecionamentos tanto de entrada quanto de saída.
5. Implementação de pipes entre programas. Você deve considerar um número arbitrário de pipes na linha de comando.
6. Os programas a serem executados devem ser pesquisados nos diretórios listados na variável de ambiente `MYPATH`. O conteúdo inicial da variável `MYPATH` deve ser importado da variável de ambiente `PATH`.
7. Implementação do comando `export` para (re)definição de variáveis de ambiente. Por exemplo, `export MYPATH=$MYPATH:/NovoDiretorio`
8. Implementação do comando `cd` para mudança de diretório. Inicialmente, o diretório corrente deve ser definido pela variável de ambiente `PWD`. Note que após a execução do comando `cd`, você deve atualizar o conteúdo da variável `PWD`.
9. Implementação do comando `history` que mostra os últimos 50 comandos digitados pelo usuário.

10. O seu interpretador de comandos deve mostrar um prompt inicial igual a `tecii$`, mas deve permitir que o prompt seja redefinido pela variável de ambiente `MYPS1`.
11. Implementação do comando `exit` para encerrar o interpretador de comandos. O interpretador também deve ser encerrado com a digitação de Ctrl+D.
12. Implementação de um tratador de sinais para tratar o sinal SIGINT e evitar que o interpretador de comandos seja terminado quando as teclas Control+C forem pressionadas.
13. Implementação de um tratador de sinais para tratar o sinal SIGTSTP e colocar para dormir o processo em *foreground*.
14. Implementação do comando `kill` para enviar sinais para processos. Note que esse deve ser um comando do interpretador e não uma invocação ao programa `/bin/kill`.
15. Implementação do comando `jobs` para listar os processos em *background*.
16. Implementação do comando `fg` para colocar um processo em *foreground*.
17. implementação do comando `bg` para colocar um processo em *background*.
18. Implementação de um tratador de sinais para tratar o sinal SIGCHLD e evitar a criação de processos zumbis.
19. Implementação do comando `echo` para visualização do conteúdo de variáveis de ambiente e impressão de cadeias de caracteres. Por exemplo, `echo $MYPATH` mostra o conteúdo da variável de ambiente `MYPATH`
20. Implementação do comando `set` para visualizar todas as variáveis de ambiente.

Utilize o comando `man` de Linux para entender os detalhes de funcionamento dos comandos listados acima. Estude também a *man page* de `signal` para entender como sinais podem ser tratados em Unix.

**Funcionalidade opcional:** implementar uma linguagem para criação e processamento de scripts. Você pode usar uma linguagem já existente como, por exemplo, de bash ou tcsh. Esta funcionalidade adicionará dois pontos ao trabalho se for corretamente implementada.

# 2 Entrega do Trabalho

O trabalho pode ser feito em grupo de no máximo três alunos e deve ser entregue até o dia 21 de maio de 2021. Além do código fonte devidamente comentado, o grupo deve entregar um breve relatório descrevendo o trabalho. Neste relatório, o grupo deve incluir uma breve introdução, decisões de implementação, funcionalidades não implementadas, problemas enfrentados na implementação, etc. O relatório deve ser entregue em um arquivo PDF. Tanto o relatório quanto os arquivos fontes da implementação devem ser colocados em um arquivo ZIP e enviados via AVA. Arquivos em outros formatos, como RAR, não serão considerados e o trabalho não será avaliado. Inclua em seu arquivo ZIP apenas os arquivos fontes e Makefile, ou seja, não inclua arquivos compilados (.o).

# 3 Avaliação

Além da correção do programa, o professor e/ou assistente de ensino poderão fazer perguntas durante uma apresentação do trabalho. Durante a apresentação, o grupo deverá explicar o funcionamento do programa e responder a perguntas relativas ao projeto.