HAUTE ÉCOLE DE
NAMUR-LIÈGE-LUXEMBOURG
**Catégorie TI/IR**
**Année académique : 2022-2023**

# Programmation orientée OS

## Consignes

La durée de l'examen est de 2 heures à répartir à votre convenance entre la partie bash et la partie PowerShell. Notez que, pour valider la matière, les acquis de base des deux parties doivent être validés.

Toute tentative de triche et/ou de plagiat se verra sanctionner d'un 0/20 et pourra faire l'objet d'un dossier auprès du conseil de discipline.

Veillez à déposer vos scripts dans les sections prévues à cet effet sur moodle.

Tous les documents, notes et/ou configurations personnelles sont autorisés. Il est également conseillé de travailler sur VMs afin de ne pas impacter votre machine physique.

Pour être recevables, vos scripts doivent s'exécuter sans erreur d'interprétation ou de syntaxe.

## Powershell

L'objectif de cette partie est de comprendre et d'expliquer le fonctionnement du script suivant :

```
1  [CmdletBinding()]
2  [OutputType()]
3  param(
4      [Parameter(Mandatory = $true)]
5      [string]$LogPath,
6      [Parameter(Mandatory = $true)]
7      [string]$ZipPath,
8      [Parameter(Mandatory = $true)]
9      [string]$ZipPrefix,
10     [Parameter(Mandatory = $false)]
```

```
    [double]$NumberOfDays = 30
)

Function Set-ArchiveFilePath{
    [CmdletBinding()]
    [OutputType([string])]
    param(
        [Parameter(Mandatory = $true)]
        [string]$ZipPath,
        [Parameter(Mandatory = $true)]
        [string]$ZipPrefix,
        [Parameter(Mandatory = $false)]
        [datetime]$Date = (Get-Date)
    )

    if(-not (Test-Path -Path $ZipPath)) {
        New-Item -Path $ZipPath -ItemType Directory | Out-Null
        Write-Verbose "Created folder '$ZipPath'"
    }

    $ZipName = "$($ZipPrefix)$($Date.ToString('yyyyMMdd')).zip"
    $ZipFile = Join-Path $ZipPath $ZipName

    if(Test-Path -Path $ZipFile) {
        throw "The file '$ZipFile' already exists"
    }

    $ZipFile
}

Function Remove-ArchivedFiles {
    [CmdletBinding()]
    [OutputType()]
    param(
        [Parameter(Mandatory = $true)]
        [string]$ZipFile,
        [Parameter(Mandatory = $true)]
        [object]$FilesToDelete,
        [Parameter(Mandatory = $false)]
        [switch]$WhatIf = $false
    )

```

```powershell
    $AssemblyName = 'System.IO.Compression.FileSystem'
    Add-Type -AssemblyName $AssemblyName | Out-Null

    $OpenZip = [System.IO.Compression.ZipFile]::OpenRead($ZipFile)
    $ZipFileEntries = $OpenZip.Entries

    foreach($file in $FilesToDelete) {
        $check = $ZipFileEntries | Where-Object{ $_.Name -eq $file.Name -and
            $_.Length -eq $file.Length }
        if($null -ne $check) {
            $file | Remove-Item -Force -WhatIf:$WhatIf
        }
        else {
            Write-Error "'$($file.Name)' was not find in '$($ZipFile)'"
        }
    }
}

$Date = (Get-Date).AddDays(-$NumberOfDays)
$files = Get-ChildItem -Path $LogPath -File |
    Where-Object{ $_.LastWriteTime -lt $Date}

$ZipParameters = @{
    ZipPath = $ZipPath
    ZipPrefix = $ZipPrefix
    Date = $Date
}
$ZipFile = Set-ArchiveFilePath @ZipParameters

$files | Compress-Archive -DestinationPath $ZipFile

$RemoveFiles = @{
    ZipFile = $ZipFile
    FilesToDelete = $files
}
Remove-ArchivedFiles @RemoveFiles

```

# Bash

## Sujet

L'objectif de cet exercice est de développer un analyseur de logs visant à surveiller facilement un ensemble de fichiers. Le programme accepte au moins arguments positionnels :

1. `<log_list_path>` : un chemin vers un fichier contenant les logs à surveiller. Chaque ligne du fichier est un chemin vers un fichier de log.
2. `<pattern>` : un pattern à rechercher.

Si plus de deux arguments positionnels sont fournis au script, tous les arguments surnuméraires sont considérés comme des patterns à rechercher dans les fichiers de log.

Pour chaque fichier de log présent dans le fichier `log_list_path`, le script recherche et affiche les occurrences de `pattern` trouvées dans le fichier. Si aucune occurrence n'est trouvée, le message `<log_file_path>: No occurence found` est affiché sur la sortie d'erreur du script.

Lorsqu'au moins une occurrence est trouvé dans un fichier, le nom du fichier est imprimé suivi de l'ensemble des lignes matchant le pattern. En d'autres termes, les occurrences doivent suivre le format suivant :

```
<log_file_path>:
<matching_line>
<matching_line>
...
```

De plus le script doit accepter les options suivantes :

`-h` : Affiche un message d'aide décrivant les options du script.

`-o <out_file>` : Enregistre les occurrences trouvées dans le fichier `out_file` au lieu de les afficher sur la sortie standard. Si aucune occurrence n'est trouvée, le message d'échec est ajouté au fichier mais est toujours affiché sur la sortie d'erreur.